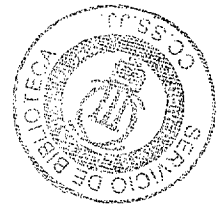
Universidad Carlos III de Madrid
Departamento de Informática

Tesis Doctoral

Selección diferenciada del conjunto de entrenamiento en Redes de Neuronas mediante aprendizaje retardado

Autor: José María Valls Ferrán
Directores:
Pedro Isasi Viñuela, Inés María Galván León
Leganés, 2004



UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

SELECCIÓN DIFERENCIADA DEL CONJUNTO DE ENTRENAMIENTO EN REDES DE NEURONAS MEDIANTE APRENDIZAJE RETARDADO

TESIS DOCTORAL

José María Valls Ferrán
Leganés, 2004

Departamento de Informática

Escuela Politécnica Superior
Universidad Carlos III de Madrid

Selección diferenciada del conjunto de entrenamiento en Redes de Neuronas mediante aprendizaje retardado

AUTOR: José María Valls Ferrán

DIRECTORES:
Pedro Isasi Viñuela, Inés María Galván León
Leganés, 2004

A Ana, Laura y Javier

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que me han ayudado y que han hecho posible que este trabajo haya podido concluirse, y en especial:

A mis directores de tesis, Inés Galván y Pedro Isasi, por haber confiado en mí y por su dedicación y paciencia. Sé que el mencionar a los directores de tesis en el apartado de agradecimientos parece un mero formulismo, por eso quiero decir que este agradecimiento es profundo y sincero. Han estado siempre a mi lado y me han ayudado y animado mucho. En los momentos duros, ellos fueron quienes hicieron que saliera adelante.

A Daniel, porque fue con quien comencé los cursos de doctorado, y por él decidí hacer la tesis en esta Universidad. A Araceli, que siempre me ha animado y ayudado; a los compañeros de Scalab, por su apoyo. Especial recuerdo a mi compañero de fatigas Germán, que está a punto de acabar su tesis. A Cristóbal, mi compañero de despacho, que me ha tenido que aguantar estos últimos meses.

Por supuesto, un especialísimo recuerdo y agradecimiento a David y a Ricardo que han compartido conmigo tantos buenos y malos momentos. También a Juan, cuántas horas de comidas y café con todos ellos charlando sobre todos los temas imaginables...

A mis compañeros del Instituto, en especial a Pilar Ortiz. Ella siempre ha estado pendiente de la marcha de mi tesis y me ha dado ánimos para continuar. A Pilar Fernández, Juan Sánchez, Juan Limón, Manuel y todos los demás que se han interesado por mi trabajo.

A mis padres y hermanos, que siempre han estado conmigo. Sé lo que esto significa para ellos.

A mis hijos Laura y Javier, a los que quiero con toda mi alma. Aunque con sus 7 y 6 años todavía no saben lo que es una tesis, sí saben que su padre ha tenido que estar demasiado tiempo pegado al ordenador estos últimos años.

Y por último, el agradecimiento más profundo para Ana. A ella debo el poder acabar esta tesis. Siempre me ha dado ánimos para seguir, aún cuando ella ha sido la más perjudicada por mi dedicación a este trabajo. A Ana, Laura y Javier espero poderles compensar por el tiempo que no he podido estar con ellos.

José María Valls
Diciembre de 2003

Resumen

Las Redes de Neuronas de Base Radial (RNBR) son aproximadores universales, en el sentido de que son capaces de aproximar, con el grado de precisión deseado, cualquier función continua multivariable, siempre que dispongan de un número suficiente de unidades ocultas. Estas redes se caracterizan por poseer características locales, ya que sus neuronas utilizan funciones de activación cuyo valor decrece exponencialmente al alejarse el patrón de entrada de sus centros. Las RNBR son modelos robustos frente a los errores en los datos y su entrenamiento es muy rápido en comparación con otros tipos de redes de neuronas.

El principal inconveniente de las RNBR reside en su deficiente capacidad de generalización. Esto se debe a que es necesario un gran número de neuronas ocultas para poder construir una aproximación a la función objetivo mediante la suma de aproximaciones locales, especialmente si la dimensión del espacio de entrada es alta; este elevado número de neuronas ocultas puede influir negativamente en la capacidad de generalización. Se ha comprobado que el nivel de generalización de las redes de neuronas depende significativamente de la calidad de los datos de entrenamiento, y algunos de esos datos pueden ser redundantes o irrelevantes. Con una cuidadosa selección de los patrones de entrenamiento se podría mejorar la capacidad de generalización.

Por otra parte, los métodos de aprendizaje retardado o "perezoso" pueden tener una buena capacidad de generalización pues construyen las representaciones de la función objetivo de forma local dependiendo de la nueva muestra de test, pero su precisión en la generalización depende significativamente del número de patrones que se seleccionen y de la función de distancia utilizada.

El objetivo principal de esta tesis consiste en mejorar la capacidad de generalización de las RNBR utilizando un enfoque basado en los métodos de aprendizaje retardado. Para ello, se propone un método de aprendizaje que selecciona automáticamente, del conjunto de entrenamiento, los patrones más apropiados para aproximar cada nueva muestra de test. Este método sigue una estrategia de aprendizaje perezoso, en el sentido de que construye aproximaciones locales centradas alrededor de la nueva muestra.

También se pretende que este método sea general, aplicable independientemente del modelo de red de neuronas elegido; de este modo, se podrá aplicar a otros tipos de redes, como el perceptron multicapa.

Para evaluar el modelo propuesto, se aplica a diferentes dominios que son representativos de problemas de aproximación de funciones, de predicción de series temporales y de clasificación. Los resultados obtenidos se comparan con los de los métodos de entrenamiento tradicionales, donde se entrenan las RNBR con todas las muestras de entrenamiento disponibles.

Abstract

Radial Basis Neural Networks (RBNN) are universal approximators, in the sense that they are able to approximate arbitrarily well any continuous multivariate function, if enough hidden units are provided. These networks have local characteristics, since their neurons use activation functions whose value exponentially decreases when the input pattern moves away from its centers. RNBR are robust models and its convergence is very fast compared to other neural models.

A poor generalization ability is the main drawback of RNBR. A great number of hidden neurons is necessary to build an approximation to the objective function by means of the sum of local approximations, specially if the dimension of the input space is high; this high number of hidden neurons can influence negatively to the network performance. It has been shown that the level of generalization of neural networks depends on the quality of the training data, and some of those data can be redundant or irrelevant. With a careful selection of the training patterns, better generalization performance may be obtained.

On the other hand, lazy learning methods can obtain a good generalization performance because they construct local representations of the objective function depending on the new test sample, but its accuracy depends significantly on the number of selected patterns and on the distance function used.

The main goal of this thesis consists of improving the generalization ability of RNBR using a lazy learning approach. Thus, a learning method that automatically selects relevant data to answer a particular novel pattern is proposed. This method follows a lazy learning strategy, in the sense that it builds local approximations centered around the new sample.

Another goal is that the method is applicable independently of the neural model chosen; in this way, it will be possible to apply the method to other types of networks, as multilayer perceptron.

In order to evaluate the proposed model, it is applied to different domains that are representative of approximation function problems, time series prediction problems, and classification problems. Results are compared with those of the traditional RBNN training methods, where all the examples from the training set are used to train the networks.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

El campo del aprendizaje automático tiene que ver con la construcción de programas de ordenador que mejoren con la experiencia de forma automática. En los últimos años se han desarrollado aplicaciones de aprendizaje automático que han tenido gran importancia práctica, pero también ha habido avances en la teoría y en los algoritmos que constituyen los fundamentos de este campo.

Para diseñar una aplicación de aprendizaje automático es necesario elegir bien el tipo de 'experiencia', es decir el tipo de ejemplos de entrenamiento, la función objetivo que debe ser aprendida, la representación para esta función, y un algoritmo para aprender la función objetivo a partir de los ejemplos de entrenamiento. El aprendizaje implica una búsqueda en un espacio de hipótesis para encontrar la hipótesis que mejor se adapte a los ejemplos de entrenamiento disponibles y a otras restricciones incluidas a priori. Existen muchos métodos de aprendizaje que buscan en los espacios de hipótesis; se pueden utilizar reglas simbólicas, árboles de decisión, redes de neuronas, etc.

Redes de Neuronas

Las **redes de neuronas** constituyen un método práctico y general para aprender funciones objetivo a partir de ejemplos. Las redes de neuronas son muy robustas frente a los errores en los datos. Para cierto tipo de problemas que implican el aprendizaje a partir de datos del complejo mundo real, como la interpretación de imágenes, el reconocimiento de voz, etc. las redes de neuronas se encuentran entre los mejores métodos de aprendizaje conocidos. Dentro de las distintas arquitecturas de las redes de neuronas supervisadas destacan dos: el Perceptron Multicapa y las Redes de Neuronas de Funciones de Base Radial (RNBR). La principal característica de estas redes es su

capacidad de aproximación universal, en el sentido de que son capaces de establecer cualquier relación no lineal entre la entrada y la salida con el grado de precisión deseado. La diferencia más importante entre ambos tipos de redes es que el perceptron multicapa construye aproximaciones *globales* a la función no lineal que mapea el espacio de entrada en la salida, mientras que las RNBR construye aproximaciones *locales* a esta función, ya que utilizan funciones de activación que decrecen exponencialmente al alejarse la entrada del centro de la neurona. Esta característica local de las RNBR hace que el proceso de entrenamiento sea mucho más rápido que en el perceptron multicapa, pues para cada patrón de entrenamiento el número de pesos que hay que ajustar es mucho menor.

Como en cualquier área del aprendizaje automático, el problema de la **generalización** es crucial. Hay que distinguir claramente entre aprender y memorizar: la red debe aprender de un conjunto de entrenamiento, y lo aprendido debe tener aplicación fuera de éste. Se dice que una red generaliza bien, cuando calcula relaciones de entrada-salida correctas para muestras no utilizadas durante el entrenamiento.

La capacidad de generalización de una red neuronal viene determinada fundamentalmente por tres factores: el tamaño del conjunto de entrenamiento y su representatividad, la arquitectura de la red y la complejidad del problema a tratar. Obviamente, no se tiene control sobre este último punto pero sí sobre los otros dos. Se han realizado muchos trabajos para mejorar la capacidad de generalización en redes de neuronas en general, y en especial, en las redes de neuronas de base radial. Estos trabajos se pueden dividir en dos grandes grupos:

- Métodos que actúan sobre los parámetros de la red. Existen métodos sofisticados que ajustan los parámetros de las redes de base radial una vez elegida la arquitectura. Otros trabajos se centran en la búsqueda de las arquitecturas óptimas de las RNBR, utilizando para ello tres tipos de técnicas: de *construcción*, que comienzan utilizando redes con un número de neuronas pequeño y van incorporando nuevas neuronas hasta conseguir la capacidad de generalización óptima. De *poda*, consistentes en comenzar con redes complejas, para ir eliminando neuronas hasta conseguir un comportamiento óptimo. Y técnicas mixtas que consisten en una combinación de las dos técnicas anteriores.

- Métodos que actúan sobre el conjunto de entrenamiento. Otros autores han centrado su estudio en el conjunto de entrenamiento, seleccionado los ejemplos más apropiados para resolver el problema. Estos métodos, que están basados en la selección de patrones, también se

llaman métodos de selección de muestras y pueden pretender distintos objetivos dependiendo del tipo de red utilizada: en el caso del perceptron multicapa, la selección de muestras se emplea para entrenar, y consiste en encontrar el subconjunto óptimo del conjunto de entrenamiento que garantiza una buena generalización. En el caso de las redes de base radial, además de lo anterior, la selección puede aplicarse para la determinación de los centros, por lo que, desde este punto de vista, estos métodos de selección podrían incluirse dentro del grupo de métodos que actúan sobre los parámetros de la red. Entre estos últimos, caben destacar los métodos que buscan localizaciones óptimas de los centros de las neuronas, basándose en una selección de las muestras de entrenamiento más apropiadas.

Las RNBR pueden tener dificultades para lograr una adecuada capacidad de generalización. En general, para poder construir una aproximación a la función objetivo mediante la suma de aproximaciones locales, se requiere un número grande de unidades ocultas, especialmente cuando el número de dimensiones del espacio de entrada es alto, y ésto puede influir negativamente en la capacidad de generalización. Son necesarios algoritmos sofisticados para lograr que aumente su precisión en la generalización.

Métodos de aprendizaje retardado

La mayor parte de los métodos de aprendizaje supervisado se pueden considerar métodos *eager* o métodos de aprendizaje temprano, donde se construye una descripción general explícita de la función objetivo, mediante los ejemplos de entrenamiento proporcionados. Cuando se recibe un patrón de test, los métodos tempranos ya han elegido su aproximación global. Esa aproximación global sobre los datos de entrenamiento que representan el dominio puede conducir a pobres resultados en la generalización.

Un enfoque alternativo que trata de mejorar la capacidad de generalización, consiste en retardar la fase de generalización hasta el momento en que se recibe una nueva muestra de test, utilizando una selección de datos de entrenamiento en lugar de utilizar todo el conjunto disponible, en el cual podría haber información irrelevante o redundante. En estos métodos, denominados métodos *lazy* o métodos de aprendizaje retardado, se construyen representaciones locales de la función objetivo cada vez que se recibe una nueva muestra de test, postponiendo la generalización hasta el momento en que se recibe la nueva muestra.

La capacidad de generalización de los métodos de aprendizaje retardado depende de la correcta elección del número de patrones a seleccionar. En

general, la precisión de la generalización aumenta con el número de patrones seleccionados, alcanza un máximo y luego decrece. El valor óptimo de este número varía según los problemas e incluso, para el mismo problema, de la zona del espacio de entrada donde se encuentre la nueva muestra. Esto es un inconveniente de los métodos retardados, pues para lograr una buena capacidad de generalización habría que determinar el número correcto de patrones a seleccionar para cada problema, e incluso, para cada muestra de test dentro de cada problema. Existen otros inconvenientes de estos métodos, como la existencia de atributos irrelevantes y la función de distancia utilizada así como su elevado coste computacional, pues para cada nueva muestra es necesario construir una aproximación local.

1.1 Objetivos de la tesis

Los objetivos de esta tesis doctoral, que se detallan en el capítulo 3, consisten en afrontar el problema de la generalización en las RNBR proponiendo métodos de entrenamiento con un planteamiento próximo al de los métodos de aprendizaje retardado.

Las RNBR son buenos modelos de aprendizaje automático, son robustas y rápidas en el entrenamiento, pero tienen el inconveniente de su deficiente capacidad de generalización, especialmente cuando el número de dimensiones del espacio de entrada es elevado. Por otra parte, los métodos de aprendizaje retardado poseen una buena capacidad de generalización, pero sólo si el número de patrones seleccionados es el adecuado. Necesitan adecuadas funciones de distancia de forma que se compense la importancia relativa de los atributos de los patrones y además, su coste computacional es elevado.

En esta tesis se pretende aprovechar las ventajas de ambos modelos utilizando una aproximación de aprendizaje retardado para entrenar RNBR, seleccionando para cada nueva muestra de test los patrones de entrenamiento más adecuados, de forma que se aumente la capacidad de generalización de estas redes, sin que sea preciso determinar el número de patrones seleccionados y de forma que la propia capacidad de ajuste de las redes solucione el problema de la distinta importancia relativa de los diferentes atributos.

Este enfoque es muy apropiado para las redes de neuronas de base radial, por la gran rapidez de su entrenamiento frente a otros modelos de redes de neuronas; pero además, se pretende conseguir que sea de aplicación general, aplicable independientemente del modelo de redes de neuronas elegido.

1.2 Organización de la tesis

El resto de los capítulos de esta memoria están organizados del siguiente modo: En el capítulo 2 se revisan los dos temas que están directamente relacionados con la presente tesis, las RNBR y los métodos de aprendizaje retardado. Se describen los principales trabajos relacionados con ellos y se finaliza con una discusión sobre los principales inconvenientes de estas técnicas.

En el capítulo 3 se exponen los objetivos de la tesis, que pretenden solucionar los inconvenientes detectados en las técnicas antes mencionadas.

En el capítulo 4, denominado *Selección de patrones de entrenamiento mediante vecindad ponderada*, se describen diferentes propuestas del método de entrenamiento de RNBR por aprendizaje retardado, y se presentan los experimentos realizados en diferentes dominios que muestran la validez del mismo. En primer lugar se describen los dominios utilizados y se muestran los resultados obtenidos cuando se les aplican RNBR entrenadas por el procedimiento convencional. A continuación se describen dos propuestas diferentes del método: la *ponderación gaussiana* y la *ponderación inversa* y se presentan los experimentos realizados con cada una de ellas.

Por último, en el capítulo 5 se muestran las principales conclusiones derivadas de este trabajo, y se plantean líneas futuras de investigación.

Capítulo 2

Estado del arte

Las Redes de Neuronas de Base Radial son sistemas pasivos de aprendizaje, porque reciben información sobre el dominio del problema e intentan ajustar los pesos para aprender las muestras de entrenamiento. Aunque se han utilizado con mucho éxito en un gran número de aplicaciones, tienen ciertos problemas inherentes al conjunto de datos de entrenamiento que utilizan para aprender.

El nivel de generalización, es decir, la capacidad para responder correctamente a nuevas entradas que no han sido vistas en el entrenamiento, depende mucho de la calidad de los datos de entrenamiento. Tradicionalmente, la red se entrena con todas las muestras disponibles sobre el dominio, que deben representar el problema de la forma más amplia posible.

Como se ha comentado en la introducción, hay muchos trabajos orientados a mejorar la capacidad de generalización de las redes de neuronas de base radial, y estos trabajos pueden agruparse en dos líneas fundamentales, los que actúan sobre la arquitectura de la red y los que actúan sobre el conjunto de entrenamiento.

Por otra parte, la mayor parte de los métodos de aprendizaje supervisado se pueden considerar métodos de aprendizaje temprano, donde se construye una descripción general explícita de la función objetivo, mediante los ejemplos de entrenamiento proporcionados. Existen unos métodos denominados métodos *lazy* o de aprendizaje retardado, donde se construyen representaciones locales de la función objetivo cada vez que una nueva muestra de test se recibe. Es decir, la generalización se postpone hasta el momento en que se recibe una nueva muestra.

En la primera sección de este capítulo se hace una revisión de las redes de neuronas de base radial, sus fundamentos teóricos y algunos de los métodos

utilizados para su entrenamiento, desde los métodos clásicos a los métodos alternativos propuestos en la literatura. También se trata el problema de la selección del modelo o cómo obtener una RNBR de un tamaño adecuado para un determinado problema.

En la segunda sección se revisan los métodos de aprendizaje retardado o métodos *perezosos* realizando a continuación una discusión sobre las RNBR y los métodos de aprendizaje retardado, sintetizando los problemas que tienen cada uno de los modelos.

2.1 Redes de Neuronas de Base Radial

El proceso de aprendizaje puede verse como el proceso de buscar en un espacio n-dimensional una superficie que se ajuste de la 'mejor manera posible' a los datos (puntos del espacio) de entrenamiento. La generalización será equivalente a la utilización de dicha superficie para interpolar los datos de test. Desde hace tiempo se ha utilizado un tipo especial de funciones llamadas Funciones de Base Radial (RBF) para resolver problemas de interpolación en espacios multidimensionales. Las RBF fueron introducidas por vez primera en la solución de este tipo de problemas -problemas de interpolación de funciones reales multivariables- por Powell [Powell, 1985] y más tarde por Broomhead [Broomhead and Lowe, 1988]. Existen trabajos más recientes sobre este importante problema [Light, 1992a], siendo uno de los campos de investigación más importantes del análisis numérico.

Las RBF se caracterizan por tener una salida que es simétrica respecto a un *centro* asociado. Es decir, $\phi_c(\mathbf{x}) = \phi(\|\mathbf{x} - \mu_c\|)$, donde ϕ es la función, μ_c el centro asociado y $\|.\|$ es una norma vectorial. Además, su salida decrece, o crece, de forma monótona con la distancia al centro. Si se utiliza la norma euclídea y se hace que $\phi(r) = \exp(-\frac{r^2}{\sigma^2})$, entonces se tiene la función Gaussiana como una función de base radial, dependiente de dos parámetros: su centro μ_c y su radio o anchura σ. En la figura 2.1 puede verse la representación de una RBF gaussiana con centro 0 y radio 1.

Una Red de Neuronas de Base Radial (RNBR), en su forma básica, no es más que la "implementación", como una red de neuronas de conexiones hacia adelante, de una combinación lineal de varias funciones de base radial.

$$y(\mathbf{x}) = \sum_{j=1}^{m} w_j \phi(\|\mathbf{x} - \mu_j\|) \tag{2.1}$$

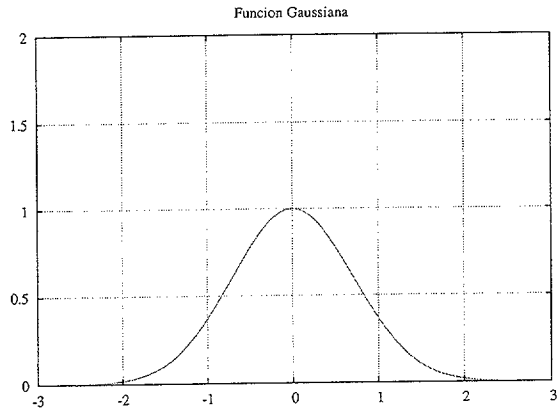


Figura 2.1:
Representación de una función de base radial unidimensional, con centro 0 y radio 1

Las RNBR, [Moody and Darken, 1989], [Poggio and Girosi, 1990], constan de tres capas: la capa de entrada, la capa oculta y la capa de salida. La capa de entrada se compone de unidades "sensoriales" que conectan la red con el espacio de entrada. La segunda capa, la capa oculta, aplica una transformación no lineal del espacio de entrada al espacio "oculto", siendo este espacio oculto de alta dimensionalidad habitualmente. Esta capa oculta se compone de unidades o neuronas ocultas, cada una de las cuales representa una función de base radial, con su centro y anchura asociada. La capa de salida realiza una combinación lineal de las activaciones de las unidades de la capa oculta, es decir, de las funciones de base radial representadas en la capa oculta.

Existe un trabajo de Cover [Cover, 1965] que permite justificar matemáticamente el uso de transformaciones no lineales seguidas de transformaciones lineales para problemas de clasificación de patrones. Según este trabajo, un problema de clasificación de patrones representado en un espacio altamente dimensional tiene más probabilidades de ser linealmente separable que si se representa en un espacio con pocas dimensiones. Esta es la razón por la que habitualmente el número de unidades de la capa oculta de una RNBR es alto. Además, la dimensionalidad del espacio oculto está directamente relacionada con la capacidad de la red de aproximar una correspondencia o "mapeado" de la entrada a la salida [Mhaskar, 1996], [Niyogi and Girosi, 1996], de forma que cuantas más neuronas ocultas, más precisa será la aproximación conseguida. Las RNBR son aproximadores uni-

versales, en el sentido de que pueden aproximar cualquier función continua sobre $\mathbf{R}^n$ con el grado de precisión deseado, propiedad que fue demostrada por Park y Sandberg [Park and Sandberg, 1991], [Park and Sandberg, 1993].

Debido a la posibilidad de la estimación por separado de los centros y anchuras de las funciones por una parte y de los pesos de las conexiones con la capa de salida por otra, se han desarrollado prácticos y rápidos algoritmos de entrenamiento para este tipo de redes.Las RNBR han adquirido una gran popularidad en los últimos años y son, quizá, el segundo tipo de redes de neuronas más utilizadas después del perceptrón multicapa, habiéndose aplicado en gran variedad de problemas. En [Haykin, 1999] se puede encontrar una amplia discusión sobre las RNBR.

El resto de la sección está organizado del siguiente modo: se tratarán los fundamentos teóricos de las RNBR, tratando el problema de la interpolación exacta y de la aproximación de funciones, viendo que no interesa aproximar con exactitud los puntos utilizados en el entrenamiento, sino generalizar con la máxima precisión posible. Se proponen como solución al problema de la interpolación exacta las redes regularizadas [Poggio and Girosi, 1990] que obtienen soluciones óptimas para aproximar funciones pero tienen el inconveniente de que necesitan tantas neuronas como puntos haya en el conjunto de entrenamiento. A continuación, se estudian las redes generalizadas que proporcionan soluciones subóptimas, utilizando un menor número de neuronas. Una vez vistos los fundamentos teóricos, se revisan los diferentes métodos utilizados para el aprendizaje de las RNBR.

2.1.1 Fundamentos teóricos de las Redes de Neuronas de Base Radial

El problema de la Interpolación Exacta

El problema de la interpolación exacta se puede formular de la siguiente forma: Dado un conjunto de N vectores o puntos diferentes de un espacio de dimensión p, $\{\mathbf{x}_i \in \mathbf{R}^p, i = 1, 2 \ldots N\}$ y un conjunto de N valores reales $t_i \in \mathbf{R} | i = 1, 2 \ldots N\}$, encontrar una función continua $y : \mathbf{R}^p \rightarrow \mathbf{R}$ tal que:

$$y(\mathbf{x}_i) = t_i, i = 1, 2, \ldots N \tag{2.2}$$

Se puede encontrar la solución a este problema utilizando las funciones de base radial de la siguiente forma [Powell, 1988]:

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{2.3}$$

donde $\phi(\|\mathbf{x} - \mathbf{x}_i\|)$ son un conjunto de N funciones de base radial cuyos centros son los N puntos $\mathbf{x}_i$ del espacio de dimensión p. Los coeficientes w_i pueden determinarse aplicando a la función 2.3 la condición del problema de interpolación especificada en la ecuación 2.2. De esta forma se obtiene un sistema de N ecuaciones lineales:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix} \tag{2.4}$$

donde

$$\phi_{ji} = \phi(\|\mathbf{x}_j - \mathbf{x}_i\|), j, i = 1, 2, \ldots, N \tag{2.5}$$

Esta ecuación matricial se puede representar en forma compacta de la siguiente forma:

$$\mathbf{\Phi w} = \mathbf{t} \tag{2.6}$$

Si la matriz $\mathbf{\Phi}$ es no singular, entonces se puede resolver la ecuación 2.6 dado que existe la matriz inversa $\mathbf{\Phi}^{-1}$. En [Michelli, 1986] y [Light, 1992b] se muestra que existe una amplia clase de funciones que cumplen la propiedad de que $\mathbf{\Phi}$ es no singular, y por tanto, con estas funciones se puede resolver la ecuación 2.6 para conseguir la interpolación exacta de los N puntos $\mathbf{x}_i$ por la función y y cuya solución viene dada por:

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{t} \tag{2.7}$$

Sustituyendo en 2.3 los pesos obtenidos en la ecuación 2.7 se obtiene la función y que representa una superficie continua en un espacio de dimensión $p + 1$ que pasaría exactamente por todos los puntos $\mathbf{x}_i$.

Entre las funciones que cumplen la propiedad citada anteriormente se encuentran las siguientes:

1. Función Gaussiana

$$\phi(r) = exp(-\frac{r^2}{2\sigma^2}), \text{ para } \sigma > 0, r \in \mathbf{R} \tag{2.8}$$

2. Función Multicuadrática

$$\phi(r) = \sqrt{r^2 + c^2}, \text{ para } c > 0, r \in \mathbf{R} \tag{2.9}$$

3. Función Multicuadrática Inversa

$$\phi(r) = \frac{1}{\sqrt{r^2 + c^2}}, \text{ para } c > 0, r \in \mathbf{R} \tag{2.10}$$

Las funciones Gaussiana y Multicuadrática inversa tienen carácter *local*, en cuanto a que su valor decrece de forma monótona con la distancia a su centro, ($\phi(r) \to 0$ si $r \to \infty$), mientras que la función Multicuadrática tiene carácter *global*, al crecer su valor tendiendo a ∞ cuando crece la distancia al centro, ($\phi(r) \to \infty$ si $r \to \infty$)

Aproximación de funciones

En la práctica, la interpolación exacta de funciones no es una buena opción pues tiene dos graves inconvenientes:

- Deficiente capacidad de generalización. Los datos generalmente tienen ruido y, por tanto, al hacer que la función de interpolación pase exactamente por todos los puntos se producirá un sobreajuste, originándose una disminución en la capacidad de generalización. Se podría lograr una mejor generalización utilizando un ajuste más suave, sin obligar a la función a pasar exactamente por cada punto; de esta forma, se promediaría el ruido de los datos.

- Alto coste computacional. Como el número de funciones de base radial necesarias para lograr la interpolación exacta debe ser igual al número de puntos, según se vio en el apartado anterior, cuando el conjunto de datos sea muy grande será muy costoso evaluar la función de interpolación.

Es mucho más interesante buscar una aproximación de funciones en lugar de una interpolación exacta, para evitar los problemas citados. Se proponen dos soluciones, la primera consiste en regularizar la solución utilizando las llamadas Redes Regularizadas [Poggio and Girosi, 1990], y la segunda en reducir el número de funciones de base radial como se puede ver en los trabajos de [Broomhead and Lowe, 1988] y [Moody and Darken, 1989]. A continuación se explican estas soluciones:

Redes Regularizadas

La idea básica de la regularización (en [Tikhonov and Arsenin, 1977] se desarrolla ampliamente la teoría de regularización de Tikhonov) consiste en

favorecer ciertas soluciones añadiendo una penalización a la función original de coste. Esta penalización aplica el principio conocido como *La Navaja de Occam* que se podría reformular de la siguiente forma: "Las soluciones deben mantenerse lo más simples posibles hasta el momento en que se demuestre que resultan inadecuadas". Es decir, se asumirá que la función de "mapeado" entre la entrada y la salida es simple o *suave*, de forma que a puntos cercanos en la entrada corresponden salidas también cercanas, y se penalizan las soluciones complejas para favorecer las soluciones simples. En [Friedman, 1994] se estudian diferentes funciones de penalización que pueden ser utilizadas para regularizar las soluciones.

Si se dispone del siguiente conjunto de datos de entrada-salida: Vectores de entrada: $\{\mathbf{x}_i \in \mathbf{R}^p, i = 1, 2 \ldots N\}$ y Valores de salida: $\{t_i \in \mathbf{R}^1 | i = 1, 2 \ldots N\}$
(Aunque se utilizan valores reales en la salida, se puede extender a un espacio n-dimensional sin pérdida de generalidad) y la función de aproximación es $y : \mathbf{R}^p \rightarrow \mathbf{R}$, basándose en la teoría de regularización de Tikhonov, esta función y se determina minimizando un funcional de coste (el funcional de coste realiza un "mapeado" de funciones -pertenecientes a un espacio de funciones adecuado- a números reales). Este funcional de coste tiene dos términos:

$$E[y] = E_s[y] + \lambda E_c[y] \tag{2.11}$$

El término $E_s[y]$ mide el error estándar entre la salida real de la función $y(\mathbf{x_i})$ y la salida deseada t_i para todos los puntos:

$$E_s[y] = \frac{1}{2}\sum_{i=1}^{N}(t_i - y(\mathbf{x}_i))^2 \tag{2.12}$$

El término $E_c[y]$ se denomina término de regularización y depende de las propiedades geométricas de la función de aproximación:

$$E_c[y] = \frac{1}{2}\|\mathbf{P}y\|^2 \tag{2.13}$$

siendo $\mathbf{P}$ un funcional que impone alguna restricción a la solución, haciendo que la función y sea suave. El parámetro λ de la ecuación 2.11 es un número real positivo llamado parámetro de regularización y sirve para controlar la influencia del término $E_c[y]$ en la solución final. Por tanto, la ecuación 2.11 queda como sigue:

$$E[y] = \frac{1}{2}\sum_{i=1}^{N}(t_i - y(\mathbf{x}_i))^2 + \lambda\frac{1}{2}\|\mathbf{P}y\|^2 \tag{2.14}$$

De esta forma, se trata de encontrar la función $y(\mathbf{x})$ que minimiza el coste definido por la ecuación 2.14, donde el primer término representa el error estándar, λ es el parámetro de regularización y $\frac{1}{2}\|\mathbf{P}y\|^2$ es el término de regularización.

Todo lo expuesto anteriormente sobre la teoría de la regularización, se puede materializar en las llamadas *redes de neuronas regularizadas*, que constan de tres capas: la primera capa o capa de entrada está compuesta por p unidades "sensoriales" o unidades de entrada, siendo p la dimensión de los datos del espacio de entrada. La segunda capa es la capa oculta, compuesta por unidades no lineales, conectadas directamente a todas las unidades de la capa de entrada. Habrá una unidad oculta por cada dato o vector de entrada $\mathbf{x_i}$ siendo $i = 1, 2, \ldots N$, siendo N el número de datos de entrada con que se entrenará la red. La capa de salida está formada por una unidad lineal conectada a todas las unidades ocultas, de forma que la salida de la red es una combinación lineal de las salidas de las unidades ocultas.

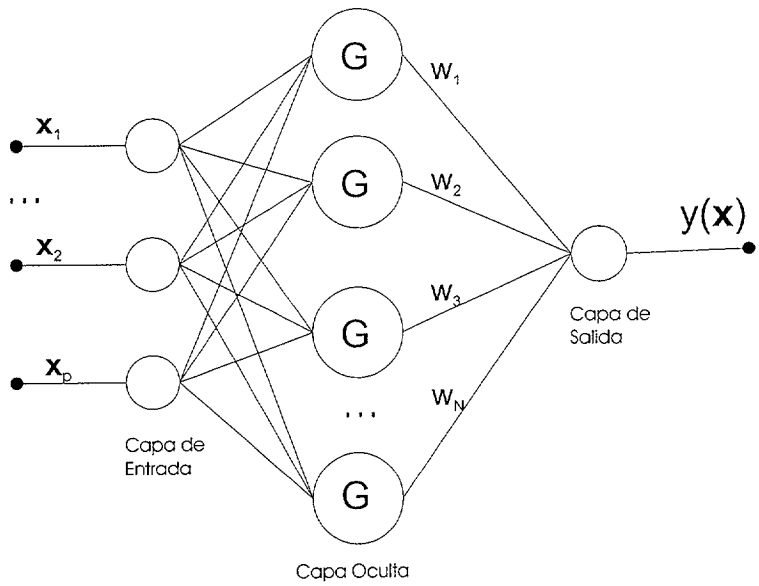


Figura 2.2: Arquitectura de la red de neuronas de base radial regularizada

Las funciones de activación de la capa oculta, deben ser funciones de Green [Poggio and Girosi, 1990], $G(\mathbf{x})$, siendo de especial interés en la práctica un subconjunto de estas funciones que son las funciones de base radial gaus-

sianas:

$$\phi(\mathbf{x}) = exp(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \mathbf{x}_i\|^2) \tag{2.15}$$

La ecuación 2.7 representaba un sistema lineal de ecuaciones para conseguir la interpolación exacta, que tenía importantes inconvenientes. En [Poggio and Girosi, 1990] se regulariza la solución utilizando el parámetro de regularización λ, de forma que el sistema de ecuaciones a resolver es:

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{w} = \mathbf{t} \tag{2.16}$$

siendo $\mathbf{I}$ la matriz identidad. Eligiendo un valor del parámetro λ lo suficientemente grande, se podrá asegurar que la matriz $(\mathbf{G} + \lambda\mathbf{I})$ es definida positiva, y por tanto existirá su inversa. De esta forma, el sistema de ecuaciones tendrá una única solución:

$$\mathbf{w} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{t} \tag{2.17}$$

Los pesos de las conexiones de la capa de salida (ver figura 2.2) son los coeficientes calculados en la ecuación 2.17, en función de las funciones de Green y el parámetro de regularización λ. El valor de la función $y(\mathbf{x})$, habiendo tomado como funciones de Greeen la función gaussiana (ecuación 2.15) es el siguiente :

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}_i\|) \tag{2.18}$$

donde los pesos w_i de las conexiones entre la capa oculta y la capa de salida son los calculados en la ecuación 2.17, dependiente del parámetro de regularización λ.
La solución proporcionada por esta red será un interpolador *óptimo*, en el sentido de que minimiza el funcional de coste especificado en la ecuación 2.11, donde existe un término que penaliza las soluciones complejas frente a las más sencillas o *suaves*. En [Poggio and Girosi, 1990] se destaca otra importante propiedad de las redes regularizadas: son *aproximadores universales*, pudiendo aproximar con el grado de precisión que se desee cualquier función continua multivariable, siempre que se disponga de un número suficiente de neuronas ocultas.

Redes Generalizadas

Aunque las redes regularizadas vistas en el apartado anterior encuentran la solución óptima, exigen disponer de tantas neuronas como puntos en el conjunto de entrenamiento, y esto puede ser inabordable en la práctica cuando

el conjunto de entrenamiento es muy grande. Para evitar el alto coste computacional requerido, se plantea un modelo, [Broomhead and Lowe, 1988], [Moody and Darken, 1989], con un número de neuronas menor, que busca una solución subóptima que aproxima la solución regularizada vista en el apartado anterior. Este modelo proporciona un ajuste más suave a los datos utilizando un número menor de funciones de base radial. Este número depende de la complejidad del problema más que del tamaño del conjunto de entrenamiento. Las modificaciones sobre el método regularizado son las siguientes:

- Número de funciones de base radial. El número de funciones, m, es típicamente menor que el número de puntos, N.
- Centros de las funciones de base radial. En lugar de situar los centros de las funciones en cada punto, ahora se determinan como parte del proceso de aprendizaje.
- Adición de umbrales o bias. Pueden añadirse umbrales a las neuronas de salida para compensar la diferencia entre el valor medio de las activaciones de las funciones sobre el conjunto de datos y los correspondientes valores de las salidas deseadas.

La ecuación correspondiente a este modelo será:

$$y(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}) + w_0 \tag{2.19}$$

donde $\phi_i(\mathbf{x})$, para $i = 1, 2, ...m$, son un conjunto de funciones de base radial, siendo típicamente $m < N$, y w_i es el conjunto de pesos. Estas funciones de base radial serán funciones gaussianas:

$$\phi_i(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{c}_i\|), i = 1, 2, ...m \tag{2.20}$$

siendo $\{c_i | i = 1, 2, ...m\}$ el conjunto de centros de las funciones, que tiene que determinarse. Queda entonces la siguiente ecuación:

$$y(\mathbf{x}) = \sum_{i=1}^{m} w_i G(\|\mathbf{x} - \mathbf{c}_i\|) + w_0 \tag{2.21}$$

Ahora deberán determinarse los pesos w_i de forma que se minimice un nuevo funcional de coste:

$$E_c[y] = \sum_{i=1}^{N}(t_i - \sum_{j=1}^{m} w_j G(\|\mathbf{x}_i - \mathbf{c}_j\|) + w_0)^2 + \|\mathbf{P}y\|^2 \tag{2.22}$$

donde el término de la izquierda corresponde al error estándar para los N puntos, y el término de la derecha corresponde al término de regularización multiplicado por el factor de regularización λ para penalizar las soluciones complejas.

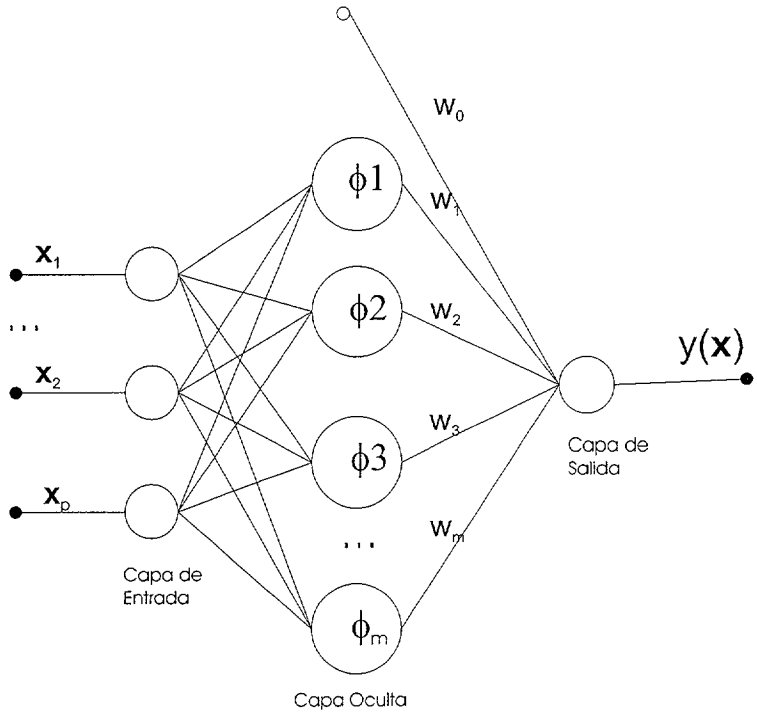


Figura 2.3: Arquitectura de la red de neuronas de base radial generalizada

Debido a que el número de funciones de base radial es menor que el número de puntos, la función es necesariamente menos compleja y el peligro de sobreadaptación a los datos de entrenamiento es menor; por tanto, puede hacerse que el segundo término sea nulo, haciendo $\lambda = 0$. En este caso, la ecuación matricial que resuelve el problema de minimización antes citado

será, [Broomhead and Lowe, 1988]:

$$\mathbf{G}^T\mathbf{G}\mathbf{w} = \mathbf{G}^T\mathbf{t} \tag{2.23}$$

Siendo $\mathbf{G}$ la matriz de funciones gaussianas:

$$\mathbf{G} = \begin{pmatrix} G(\mathbf{x}_1,\mathbf{c}_1) & G(\mathbf{x}_1,\mathbf{c}_2) & \dots & G(\mathbf{x}_1,\mathbf{c}_m) & 1 \\ G(\mathbf{x}_2,\mathbf{c}_1) & G(\mathbf{x}_2,\mathbf{c}_2) & \dots & G(\mathbf{x}_2,\mathbf{c}_m) & 1 \\ \vdots & \vdots & & \vdots & 1 \\ G(\mathbf{x}_N,\mathbf{c}_1) & G(\mathbf{x}_N,\mathbf{c}_2) & \dots & G(\mathbf{x}_N,\mathbf{c}_m) & 1 \end{pmatrix} \tag{2.24}$$

que no es simétrica sino de orden N x m. La última columna de la matriz corresponde a los umbrales de las neuronas de salida o *bias*, que equivalen a una función de base radial de valor constante e igual a 1. La matriz $\mathbf{w}$ será la matriz de pesos y $\mathbf{t}$ será la matriz de los valores de salida deseados, que por simplicidad se han supuesto unidimensionales, aunque puede generalizarse cualquier número de dimensiones:

$$\mathbf{w} = \begin{pmatrix} w_1 & w_2 & \dots & w_m & w_0 \end{pmatrix}^T \tag{2.25}$$

siendo w el umbral o bias.

$$\mathbf{t} = \begin{pmatrix} t_1 & t_2 & \dots & t_N \end{pmatrix}^T \tag{2.26}$$

Resolviendo la ecuación 2.23, la matriz de pesos será

$$\mathbf{w} = \mathbf{G}^+\mathbf{t} \tag{2.27}$$

siendo $\mathbf{G}^+$ la seudoinversa de la matriz $\mathbf{G}$:

$$\mathbf{G}^+ = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T \tag{2.28}$$

Convergencia de las RNBR

Ya que las RNBR pueden aproximar cualquier función continua con el grado de precisión deseado, siempre que dispongan del suficiente número de neuronas ocultas, es importante conocer cómo aumenta la precisión de la aproximación con el número de neuronas ocultas. En [Xu et al., 1994] se obtienen resultados interesantes para RNBR normalizadas. En este trabajo se establecen relaciones útiles entre las RNBR y KRE (kernel regression estimators). Utilizando resultados teóricos existentes sobre KRE, se obtienen interesantes resultados teóricos sobre RNBR estableciendo límites superiores

para tasas de convergencia del error de aproximación en función del número de neuronas ocultas. En concreto, para las clase de funciones que son de orden q, el límite superior es:

$$O(m^{-\frac{2q}{2q+p}})$$

siendo m el número de neuronas ocultas y p el número de dimensiones de la entrada. Por tanto, cuanto más "suave" sea la función, es decir, cuanto mayor sea q, más rápida será la convergencia. Si la relación p/q se mantiene constante, la tasa de convergencia no dependerá de la dimensionalidad de la entrada. Además, si $q >> p$ la tasa se aproxima a $O(m)$. En el trabajo citado anteriormente, no se tiene en cuenta el número de ejemplos de entrenamiento utilizados, ya que sólo se contempla el error de aproximación a estos ejemplos.

En [Niyogi and Girosi, 1996] se tiene en cuenta el error de generalización y se establecen límites a la tasa de convergencia en función del número de ejemplos de entrenamiento disponibles. En [Krzyzak and Linder, 1997] se estudian técnicas que permiten seleccionar el tamaño adecuado de la red, dependiendo del número de ejemplos de entrenamiento y la dimensión VC del problema, para minimizar el riesgo empírico, relacionado con el error de generalización.

2.1.2 Aprendizaje de las redes de neuronas de base radial

El proceso de aprendizaje de las redes de neuronas de base radial implica la determinación de todos los parámetros que intervienen en la red (ecuación 2.21): el número de neuronas ocultas m, los centros c_i las desviaciones σ_i de las neuronas ocultas y los pesos w_i de las conexiones entre la capa oculta y la capa de salida, así como los umbrales de las neuronas de salida (estos umbrales se pueden incluir en el conjunto de pesos). La elección del número de neuronas ocultas, m, se tratará con detalle en el apartado 2.1.3. Fijado este número m, el entrenamiento de una RNBR implica la determinación de los valores de tres conjuntos de parámetros -centros, desviaciones y pesos- de manera que se minimice la función de coste adecuada. Hay varias formas de entrenar las RNBR, entre las que destacan las siguientes:

Aprendizaje en una fase, ajustando sólamente los pesos

Es el proceso más simple, y consiste en ajustar sólamente los pesos **w** de la capa de salida, mediante algún tipo de optimización supervisada, por ejemplo, minimizando la diferencia cuadrática entre la salida de la red y

la salida deseada. Los centros $\mathbf{c}_i$ coincidirán con un subconjunto de los puntos de entrada, que pueden ser elegidos aleatoriamente. Este enfoque se justifica porque se supone que los datos de entrenamiento están distribuidos de forma que representan el problema. Normalmente, las desviaciones σ_i de las funciones toman el mismo valor:

$$\sigma = \frac{d_{max}}{\sqrt{2m}} \tag{2.29}$$

siendo d_{max} la distancia máxima entre los centros seleccionados y m el número de centros. De esta forma, se asegura que las funciones gaussianas cubren adecuadamente el espacio de entrada, no siendo ni demasiado planas ni demasiado estrechas. Un enfoque alternativo consiste en utilizar desviaciones más grandes en centros situados en zonas de baja densidad de patrones y desviaciones más pequeñas para centros situados en zonas de alta densidad de patrones, pero esto supone realizar aprendizaje de este parámetro experimentando con los datos del conjunto de entrenamiento.

Un tipo de esta clase de aprendizaje de una fase es el aprendizaje de vectores de soporte para redes de base radial, basado en las máquinas de vectores de soporte (MVS), donde los centros se localizan en ciertos puntos del conjunto de entrenamiento y las desviaciones son iguales para todas las funciones. Además, el número de centros se puede determinar automáticamente [Cristianini and Shawe-Taylor, 2000], [Vapnik, 1998], [Scholkopf et al., 1998].

Aprendizaje totalmente supervisado

En este tipo de aprendizaje todos los parámetros de la RNBR -centros, amplitudes, pesos y umbrales- se determinan de manera completamente supervisada, con el objetivo de minimizar una función de coste. Al utilizar este método, en ningún momento el proceso de aprendizaje se guía para que las amplitudes alcancen valores pequeños tales que el solapamiento de las activaciones de las neuronas ocultas sea lo más suave posible, sino que se determinan para minimizar la función de coste; por tanto, no debe esperarse que la red siga conservando sus características locales. La forma más natural para realizar este proceso consiste en utilizar un método de descenso de gradiente sobre la función de coste dada por la suma de los errores cuadráticos computados en la salida de la red para todos los patrones de entrenamiento. Sea E el la suma de los errores cuadráticos para todos los

patrones de entrenamiento:

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{2.30}$$

donde, para abreviar la notación, $y_i = y(\mathbf{x}_i)$ siendo el valor que toma la salida de la red para el patrón $\mathbf{x}_i$, y t_i es el valor deseado del patrón $\mathbf{x}_i$. Utilizando la técnica de gradiente descendiente para minimizar el error E, se van adaptando los centros, las desviaciones y los pesos a medida que se presenta cada patrón $\mathbf{x}_i$ siguiendo las siguientes leyes:

- **Pesos** La modificación que debe aplicarse a los pesos es la siguiente:

$$\Delta w_j = \alpha_1(t_i - y_i)\phi_j(\mathbf{x}_i) \tag{2.31}$$

  siendo Δw_j el incremento que debe aplicarse al peso de la conexión entre la unidad oculta j y la unidad de salida, cuando se le ha presentado el patrón $\mathbf{x}_i$, y α_1 la tasa de aprendizaje de los pesos. La ley de aprendizaje correspondiente a los umbrales es:

$$\Delta w_0 = \alpha_1(t_i - y_i) \tag{2.32}$$

  Se puede considerar el umbral como el peso de una conexión entre una neurona oculta cuya activación es constante e igual a 1 y una unidad de salida. Por esto, a partir de ahora no se mencionarán los umbrales, sino que se incluirán en los pesos.

- **Centros.** El incremento que hay que aplicar al vector centro de la unidad j, $\mathbf{c}_j$, cuando se ha presentado el patrón $\mathbf{x}_i$ es el siguiente:

$$\Delta \mathbf{c}_j = \alpha_2\phi_j(\mathbf{x}_i)\frac{(\mathbf{x}_i - \mathbf{c}_j)}{\sigma_j^2}(t_i - y_i)w_j \tag{2.33}$$

  donde α_2 es la tasa de aprendizaje de los centros.

- **Desviaciones** La ley que determina el incremento que hay que aplicar a la amplitud de cada neurona oculta, cuando se ha presentado a la red el vector de entrada $\mathbf{x}_i$ es la que sigue:

$$\Delta \sigma_j = \alpha_3\phi_j(\mathbf{x}_i)\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{\sigma_j^3}(t_i - y_i)w_j \tag{2.34}$$

Como se vio anteriormente, la salida de la red depende linealmente de los pesos, pero la dependencia es no lineal respecto de los parámetros de las neuronas ocultas (centros y desviaciones). Por esta razón, este proceso iterativo de descenso del gradiente encontrará un mínimo local de la función de coste que será dependiente de la inicialización de los parámetros. Generalmente, en el contexto de redes de neuronas, dicha inicialización suele hacerse de manera aleatoria y con valores cercanos a cero. Sin embargo, en el caso de las redes de base radial, ésta no tiene por qué ser la manera más adecuada de inicializar, fundamentalmente, los centros de las funciones de base radial. En este caso, sería aconsejable inicializar los centros de manera que representen zonas del espacio de entrada, limitando la búsqueda del método a ciertas regiones de ese espacio. Por ejemplo, los centros podrían inicializarse a patrones de entrada de manera aleatoria o incluso a valores alcanzados después de aplicar un algoritmo de clasificación no supervisada a los patrones de entrada, lo cual facilitará la labor de optimización al método de descenso del gradiente.

Cabe preguntarse si realmente se mejora la capacidad de generalización de las redes de base radial al permitir que los centros se adapten. En un trabajo de Lowe [Lowe, 1989] se indica que la optimización no lineal de los parámetros de las funciones de base radial son beneficiosas cuando se utiliza una configuración de red mínima, pero se puede conseguir la misma capacidad de generalización utilizando una RNBR con más neuronas ocultas con centros fijos, adaptando sólamente los pesos de las conexiones a la capa de salida en un proceso de optimización lineal. En [Wettschereck and Dietterich, 1992] se realiza el experimento conocido como *NETtalk* para comparar el rendimiento de RNBR con centros fijos con RNBR cuyos centros se han ajustado mediante aprendizaje supervisado; además se han comparado estos resultados con los obtenidos en el experimento original *NETtalk* realizado por [Sejnowski and Rosenberg, 1987]. En dicho experimento se utilizó el perceptron multicapa entrenado con el algoritmo de retropropagación, para aprender a mapear el inglés escrito en su correspondiente pronunciación fonética. Las conclusiones del experimento de Wettschereck y Dietterich son las siguientes: Las RNBR entrenadas con los centros fijos y donde se habían determinado los pesos de forma supervisada, no alcanzaban el mismo nivel de generalización del perceptron multicapa. Sin embargo, las RNBR entrenadas de forma totalmente supervisada (centros, desviaciones y pesos) superaban la capacidad de generalización del perceptron multicapa.

Aprendizaje en dos etapas

La actualización simultánea de todos los parámetros de la red (centros, desviaciones y pesos), utilizando las ecuaciones 2.31, 2.33 y 2.34 puede ser conveniente cuando los datos no están disponibles en su totalidad sino que van llegando de forma dinámica y queremos entrenar la red en línea. Sin embargo, si se dispone de todos los datos de entrenamiento, se puede separar en dos etapas el proceso de optimización de los parámetros de la capa oculta y los de la capa de salida mediante la utilización de diferentes técnicas. Así, para los parámetros de la capa oculta -centros y desviaciones- el proceso de aprendizaje debe estar guiado por una optimización en el espacio de patrones de entrada, pues cada una de las neuronas ocultas en la red de base radial va a representar una zona diferente del espacio de entrada. Sin embargo, para los parámetros de la capa de salida -los pesos- la optimización se debe realizar en base a las salidas que se desean obtener, ya que las redes de base radial se utilizan para aproximar relaciones entre el conjunto de variables de entrada y salida que definen el problema. Por tanto, uno de los mecanismos más usados para el aprendizaje de las redes de base radial es el llamado método en dos etapas o método híbrido [Moody and Darken, 1989] que consiste en dos fases:

1. Fase no supervisada para la determinación de los centros ($\mathbf{c}_j$) y las desviaciones (σ_j)

2. Fase supervisada para la determinación de los pesos (w_j).

Estas dos subtareas pueden realizarse por separado y esto permite realizar el entrenamiento de forma muy eficiente. En la primera fase solamente se utilizan los datos de entrada $\{\mathbf{x}_i\}$ para determinar los centros $\mathbf{c}_j$ y las desviaciones σ_j de las funciones de base radial, siendo, por tanto, un aprendizaje no supervisado. Una vez que los parámetros de las funciones de base radial están determinados se puede pasar a la segunda fase donde se puede emplear un aprendizaje supervisado -utilizando la información de la salida que corresponde a cada vector de entrada- para determinar los pesos de las conexiones de la capa de salida. A continuación se describen con detalle estas dos etapas.

- **Fase no supervisada: Determinación de los parámetros de las RBF**

  Las neuronas ocultas de una RNBR se caracterizan porque tienen una activación local, en el sentido de que cada neurona oculta va a re-

presentar una zona del espacio de entrada. Por tanto, los centros y las desviaciones de las funciones de base radial deben ser determinados con este objetivo, es decir con el objetivo de agrupar o clasificar el espacio de entrada en diferentes zonas o clases. El representante de cada clase será el centro de la función de base radial y su desviación vendrá dada por la amplitud de cada clase. Los centros de las funciones de base radial se determinan mediante un algoritmo de clasificación no supervisado que permita dividir el espacio de patrones de entrada en clases o grupos, siendo el representante de cada grupo el centro de cada función de base radial. El método más utilizado para determinar los grupos o clases es el algoritmo de K-medias tradicional, método utilizado por [Moody and Darken, 1989], aunque también han sido propuestas una gran variedad de técnicas de clasificación no supervisada: [Chinrungrueng and Séquin, 1995],
[Ismail and Kamel, 1989], [Lloyd, 1982], [Orr, 1996]. A continuación se hace una revisión de algunos de estos métodos.

Algoritmo de K-medias tradicional

El algoritmo tradicional de K-medias [MacQueen, 1967] es un algoritmo sencillo para clasificación o agrupamiento de datos, y divide el espacio de entrada en K clases o regiones, situando el centroide de cada clase en su centro geométrico. Las regiones se definen minimizando la suma de las distancias cuadráticas entre cada vector de entrada y el centro de su correspondiente clase. El algoritmo puede seguir dos enfoques distintos: K-medias por lotes, y K-medias en línea. El primero se aplica cuando todos los datos de entrada están disponibles desde un principio, mientras que el segundo se aplica cuando no se dispone de todos los patrones desde el primer momento, sino que pueden añadirse patrones adicionales más tarde. Cuando se aplica el algoritmo por lotes, se debe seleccionar arbitrariamente una partición inicial de forma que cada grupo disponga de, al menos, un patrón. Como la totalidad de los datos están disponibles, los centros de cada partición se calculan como la media de los patrones pertenecientes a ese grupo. A medida que el algoritmo se va ejecutando, algunos patrones cambian de una clase a otra debiendo recalcularse los centros en cada paso. El siguiente algoritmo resumido es la base de las dos modalidades mencionadas de K-medias:

1. Seleccionar de forma arbitraria una partición en K regiones inicial.

2. Calcular los centros de cada región.
3. Redistribuir los patrones entre las regiones.
4. Volver al paso 2 hasta que no se aprecie cambio en los centros de las regiones.

Mediante este algoritmo el espacio de patrones de entrada se divide en K clases o regiones, y el representante de cada una de estas clases c_j será el centro de la neurona oculta j. Dichos centros se determinan con el objetivo de minimizar las distancias cuadráticas euclídeas entre los patrones de entrada y el centro más cercano, es decir minimizando el valor J:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{N} M_{in} \|\mathbf{x}_i - \mathbf{c}_j\|^2 \tag{2.35}$$

donde N es el conjunto de patrones, $\|.\|$ es la distancia euclídea, $\mathbf{x}_i$ es el patrón de entrada i, c_j es el centro de la región j, y M_{in} es la función de pertenencia del patrón i a la región j de forma que vale 1 si el centro c_j es el más cercano al patrón x_i y 0 en caso contrario, es decir:

$$M_{in} = \begin{cases} 1 & \text{si } \|\mathbf{x}_i - \mathbf{c}_j\|^2 < \|\mathbf{x}_i - \mathbf{c}_s\|^2 \ \forall s \neq j,\ s = 1, 2, ..., K \\ 0 & \text{en otro caso} \end{cases} \tag{2.36}$$

A continuación se pasa a detallar la modalidad "por lotes" del algoritmo: Siendo K el número de clases, $\{\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})\}_{i=1...N}$ el conjunto de patrones de entrada y $\{C_j = (c_{j1}, c_{j2}, ..., c_{jp})\}_{j=1,...,K}$ los centros de las clases, los pasos detallados para la aplicación del algoritmo son los siguientes:

1. Se inicializan los centros de las K clases. Los centros podrían inicializarse a K patrones aleatorios del conjunto de patrones disponibles o bien puede realizarse aleatoriamente, en cuyo caso es conveniente que se inicialicen dentro del rango de valores de los patrones de entrada.
2. Se asignan N_j patrones de entrada a cada clase j del siguiente modo:

   El patrón $\mathbf{x}_i$ pertenece a la clase j si $\|\mathbf{x}_i - \mathbf{c}_j\|^2 < \|\mathbf{x}_i - \mathbf{c}_s\|^2 \forall s \neq j$, $s = 1, 2, ..., K$

Por tanto, cada clase tendrá asociado un determinado número de patrones de entrada, aquellos más cercanos al centro de la clase.

3. Se calcula la nueva posición de los centros de las clases como la media de todos los patrones que pertenecen a su clase, es decir:

$$c_{jk} = \frac{1}{N_j} \sum_{i=1}^{N_j} M_{in} x_{ik} \text{ para } k = 1, 2, ...p \ , \ j = 1, 2, ..., K \quad (2.37)$$

4. Se repiten los pasos 2 y 3 hasta que las nuevas posiciones de los centros no se modifiquen respecto a su posición anterior, es decir hasta que:

$$\|\mathbf{c}_j^{nuevo} - \mathbf{c}_j^{anterior}\| < \varepsilon \ \forall \ i = 1, 2, ..., K \quad (2.38)$$

siendo ε un número real positivo próximo a cero.

El algoritmo de K-medias es un método sencillo y eficiente pero tiene el inconveniente de que, dependiendo de la inicialización de los centros, suele converger hacía un mínimo local de la función J dada por la ecuación 2.35, resultando una solución no óptima.

Optimización iterativa

La técnica de la Optimización Iterativa [Duda and Hart, 1973] emplea búsqueda en anchura y examina el efecto sobre la función objetivo del movimiento de un patrón de una región a otra. Comenzando con una partición arbitraria, se examinan todos los movimientos posibles, tomándose como movimiento definitivo aquél que consigue el decremento mayor del valor de la función objetivo. Si se considera, como ejemplo, que el patrón $\mathbf{x}_i$ pertenece a la región s, sólamente se moverá a otra región r si se cumplen las siguientes condiciones:

$$\frac{n_s \|\mathbf{x}_i - \mathbf{c}_s\|^2}{n_s - 1} > \frac{n_r \|\mathbf{x}_i - \mathbf{c}_r\|^2}{n_r + 1} \quad (2.39)$$

y además

$$\frac{n_r \|\mathbf{x}_i - \mathbf{c}_r\|^2}{n_r + 1} = \min_{1 \leq j \leq k, k \neq s} \frac{n_j \|\mathbf{x}_i - \mathbf{c}_j\|^2}{n_j + 1} \quad (2.40)$$

siendo n_j el número de patrones que hay en la región j. El algoritmo es el siguiente:

1. Seleccionar una partición inicial arbitraria de las n muestras en k regiones.
2. Computar los centros $\{c_1, c_2, \ldots, c_K\}$ de cada región como las medias de las muestras.
3. Seleccionar la siguiente muestra candidata $\mathbf{x}_i$. Supóngase que $\mathbf{x}_i$ está actualmente en la región s.
4. Si $n_i = 1$ ir a paso 6, ya que las regiones con un sólo elemento no deben quedarse sin él.
   Si no, hacer:
$$p_j = \frac{n_j \|\mathbf{x}_i - \mathbf{c}_j\|^2}{n_j + 1} \text{ para } j \neq s$$
$$p_j = \frac{n_s \|\mathbf{x}_i - \mathbf{c}_j\|^2}{n_j - 1} \text{ para } j = s$$
5. Transferir $\mathbf{x}_i$ a la región r si $p_r \leq p_j, \forall j$
6. Actualizar los centros $\{c_1, c_2, \ldots, c_k\}$.
7. Si no hay cambio en la función objetivo después de un determinado número máximo de intentos,
   entonces parar; si no, ir al paso 3.

Técnica de Primero en Profundidad

La técnica de Primero en Profundidad está basada en la búsqueda en profundidad y es similar a la de optimización iterativa, habiendo sido propuesta por [Ismail et al., 1984]. Mueve un patrón de una región a otra hasta que se consigue la primera mejora en la función objetivo. El algoritmo es el siguiente:

1. Seleccionar una partición inicial arbitraria de las N muestras en K regiones.
2. Computar los centros $\{c_1, c_2, \ldots, c_K\}$ de cada región.
3. Para $l = 1$ hasta N hacer: (se asume que x_l pertenece a la región de centro c_i, siendo n_i el número de patrones de esa región).
   Si $n_i \neq 1$ entonces
$$\Delta_i = \frac{n_i \|x_i - c_i\|^2}{n_i - 1}$$

Para $j = 1$ hasta k hacer:
Si $j \neq i$ entonces:

$$\Delta_i = \frac{n_j \|x_i - c_j\|^2}{n_j + 1}$$

Si $\Delta_j < \Delta_i$ entonces:
Mover x_l a c_j
Actualizar c_i y c_j Actualizar n_i y n_j

4. Volver a paso 3 hasta que no se produzcan cambios en la función objetivo después de un determinado número máximo de intentos.

K-medias adaptativo óptimo

Como se comentó en la explicación del algoritmo K-medias tradicional, el inconveniente principal que tiene dicho método es que frecuentemente encuentra soluciones subóptimas, cayendo en mínimos locales. [Chinrungrueng and Séquin, 1995] proponen una técnica llamada K-medias adaptativo óptimo, que permite una mejora del método tradicional aproximándose a la solución óptima con una eficiente tasa de aprendizaje adaptativa. Este método consigue escapar de los mínimos locales ecualizando las variaciones v_k entre las diferentes regiones. Esta ecualización se consigue introduciendo un sesgo en la medida de distancia de forma que se favorezcan las regiones con menor variación. Se define la distancia $d(x, c_k)$ entre un patrón de entrada x y el centro c_k de la región k, como el cuadrado de la norma euclídea multiplicada por la variación dentro de la región v_k.

$$d(x, c_k) = v_k \cdot \|x - c_k\|^2 \tag{2.41}$$

Las variaciones en las regiones se determinan de forma adaptativa mediante la siguiente expresión, donde se calcula la estimación de la variación de la región k en el instante T+1, $\hat{v}_k^{T+1}$, en función de la variación en el instante T, $\hat{v}_k^T$:

$$\hat{v}_k^{T+1} = \alpha \cdot \hat{v}_k^T + (1 - \alpha) M_{k,bias}(x_T) \|x^T - c_k^T\|^2 \tag{2.42}$$

siendo α una constante de valor muy próximo a 1 que define la precisión del estimador. En el instante 0 los estimadores $\hat{v}_k^0$ se inicializan a un valor muy pequeño, de esta forma, los efectos debidos a la inicialización desaparecen rápidamente. La función de pertenencia $M_{k,bias}$ se define

de la siguiente forma:

$$M_{k,bias} = \begin{cases} 1 & \text{si } v_k\|x - c_k\|^2 \leq v_i\|x - c_i\|^2 \text{ , para } i \neq k \\ 0 & \text{, en otros casos} \end{cases} \tag{2.43}$$

Como se dijo anteriormente, la tasa de aprendizaje η es adaptativa y estará relacionada con la diferencia entre la "calidad" de la partición actual y la de la partición objetivo. Cuando la partición actual está lejos del objetivo, la tasa de aprendizaje será grande para así poder avanzar más rápidamente en cada iteración. Cuando se esté cerca del objetivo, la tasa será pequeña para avanzar con más precisión. La medida de calidad de una determinada partición depende de la similitud de las diferentes particiones v_k. Se define la calidad como la entropía de las variaciones normalizadas de las regiones:

$$H(v_1, v_2, \ldots, v_K) = \sum_{i=1}^{K}(-v_i^{norm} \cdot ln(v_i^{norm})) \tag{2.44}$$

siendo la variación normalizada de una región v_i:

$$v_i^{norm} = \frac{v_i}{\sum_{j=1}^{k} v_j} \tag{2.45}$$

Entonces, de acuerdo con esta medida de calidad, la tasa de aprendizaje η en el instante de tiempo T será:

$$\eta^T = ln(k) - H^T(v_1, v_2, \ldots, v_K) \tag{2.46}$$

Discusión sobre los diferentes métodos de determinación de los centros

En [de Carvalho and Brizzotti, 2001] se comparan los rendimientos de RNBR cuyos centros se han determinado utilizando estos métodos de clasificación no supervisada, midiendo la tasa de aciertos sobre el conjunto de test en un determinado dominio de clasificación de imágenes. Los resultados muestran que aunque las diferentes técnicas pueden producir diferentes particiones del espacio de entrada, no se destancan significativas diferencias en el rendimiento de las RNBR.

Determinación de las amplitudes

Una vez determinados los centros de las funciones de base radial, deben calcularse las amplitudes o desviaciones de dichas funciones. Según

[Bishop, 1995] la determinación de las amplitudes es un asunto crítico en el entrenamiento de las RNBR. Cuando la amplitud σ es demasiado grande, la densidad de probabilidad estimada está demasiado suavizada y se puede perder la natulareza de la verdadera densidad de probabilidad subyacente. Por el contrario, cuando σ es demasiado pequeña, puede producirse una sobreadaptación al conjunto de datos concreto. Además, las amplitudes muy grandes o muy pequeñas suelen causar problemas numéricos con los métodos de gradiente descendente.

Existen varias formas de calcular la amplitud: pueden calcularse de forma que sean idénticas para todas las funciones o bien que sean diferentes para cada función dependiendo de la parte del espacio de entrada que esta función representa. En el primer caso, la amplitud puede calcularse como un múltiplo de la distancia media entre centros. Este múltiplo determina la suavidad de las funciones: amplitudes pequeñas producen funciones poco suaves. En el caso de amplitudes diferentes para cada función, cada amplitud se puede calcular utilizando heurísticas basadas en los p-vecinos más cercanos tal y como propuso Moody ([Moody and Darken, 1989]), las cuales permiten que el solapamiento entre las neuronas ocultas sea lo más suave posible. Se pueden utilizar diferentes heurísticas, como por ejemplo:

- Media uniforme de las distancias euclídeas del centro c_i a los p centros más cercanos:

$$\sigma_i = \frac{1}{p} \sum_p \|c_i - c_p\| \tag{2.47}$$

- Otra opción bastante efectiva es determinar la amplitud de la función de base radial como la media geométrica de la distancia del centro a sus dos vecinos más cercanos:

$$\sigma_i = \sqrt{\|c_i - c_t\| \ \|c_i - c_s\|} \tag{2.48}$$

  siendo c_t y c_s los dos centros más cercanos al centro c_i.

  En [Schwenker et al., 2001] se analizan varios métodos adicionales para calcular las desviaciones. En uno de ellos, todos los σ_j toman el mismo valor σ que es proporcional a la media de las p mínimas distancias entre todos los pares de centros. En otro método, se calcula la desviación σ_j como la media de las distancias entre los datos de entrada que pertenecen a la correspondien-

te región C_j:

$$\sigma_j = \alpha \frac{1}{|C_j|} \sum_{x \in C_j} \|x - c_j\| \tag{2.49}$$

donde α es un parámetro real positivo que debe ser establecido heurísticamente y $|C_j|$ indica el número de patrones pertenecientes a la región C_j.

- **Fase supervisada**
  En esta fase se calculan los pesos de las neuronas de salida de la red. En este caso, el objetivo es minimizar las diferencias entre las salidas de la red y las salidas deseadas. Por tanto, el proceso de aprendizaje está guiado por la minimización del error cuadrático medio computado en la salida de la red para todos los patrones de entrenamiento:

$$E = \frac{1}{N} \sum_{i=1}^{N} e_i \tag{2.50}$$

donde N es el número de patrones o muestras y e_i es el error cometido por la red para el patrón $\mathbf{x}_i$, que viene dado generalmente por:

$$e_i = \frac{1}{2}(t_i - y_i)^2 \tag{2.51}$$

siendo $y_i = \mathrm{y}$ t_i los valores de la salida de la red y la salida deseada para el patrón de entrada $\mathbf{x}_i$, respectivamente. Para resolver este problema de optimización se suele utilizar una técnica basada en la corrección del error. En la ecuación (2.19), se observa que las salidas de la red de base radial dependen linealmente de los pesos de la red, por lo que un método bastante simple y eficiente de utilizar es el algoritmo de los mínimos cuadrados para minimizar el error. De este modo, los pesos de la red se determinan mediante un proceso iterativo gobernado por las siguientes leyes:

$$w_{ji} = w_{ji-1} + \alpha_1(t_i - y_i)\phi_{ji} \tag{2.52}$$

$$w_{0i} = w_{0i-1} + \alpha_1(t_i - y_i) \tag{2.53}$$

$$\text{para } i = 1, 2, ..., N \text{ y para } j = 1, ..., m$$

donde w_{ji} es el peso de la conexión entre la neurona oculta j y la salida, para el patrón de entrada $\mathbf{x}_i$, ϕ_{ji} es la activación de la neurona

oculta j para el mismo patrón de entrada, w_{0i} es el umbral para ese patrón y α_1 es la razón o tasa de aprendizaje.

Cuando se calculan los pesos mediante la ley de aprendizaje dada por las ecuaciones (2.52) y (2.53), la convergencia es bastante rápida, consiguiendo una solución en un número pequeño de ciclos de aprendizaje.

Debido a que la salida de la red depende linealmente de los pesos y umbrales, otro método para el cálculo de dichos parámetros es el llamado *método de la seudo-inversa* [Haykin, 1999] que ya se ha explicado en el apartado 2.1.1. Dicho método proporciona una solución directa al problema de optimización que viene dada por la siguiente expresión matricial:

$$\mathbf{w} = \mathbf{G}^{+} \cdot \mathbf{t} = (\mathbf{G}^{T} \cdot \mathbf{G})^{-1} \cdot \mathbf{G}^{T} \cdot \mathbf{t} \tag{2.54}$$

donde $\mathbf{w}$ es la matriz de pesos y umbrales de orden $(m+1) \times 1$ y $\mathbf{G}^{+} = (\mathbf{G}^{T} \cdot \mathbf{G})^{-1} \cdot \mathbf{G}^{T}$ es la matriz seudoinversa de $\mathbf{G}$, siendo $\mathbf{G}^{T}$ la matriz traspuesta de $\mathbf{G}$. $\mathbf{G}$ es una matriz de orden $N \times (m+1)$ que contiene las activaciones de las neuronas ocultas de la red para los N patrones de entrada. $\mathbf{t}$ es la matriz de salidas deseadas para la red, de orden $N \times 1$.

De este modo, y como se observa en la ecuación (2.54), los pesos y umbrales de la red se pueden calcular de manera directa, basta calcular la seudo-inversa de la matriz G, obteniendo así una solución óptima al problema de minimización.

Sin embargo, es necesario señalar que, aunque dicho método proporciona una solución directa al problema, desde un punto de vista práctico no es el método más eficiente, pues la solución implica el cálculo de la matriz seudo-inversa, el cual debe realizarse mediante métodos numéricos [Haykin, 1999] que podrían requerir un alto coste computacional, así como la producción de errores debido a problemas de precisión. Por tanto, en el contexto de redes de neuronas de base radial, el método más utilizado para la determinación de pesos y umbrales es el algoritmo de los mínimos cuadrados.

Otros modos de entrenamiento de las RNBR

Como una variación de los métodos en dos fases vistos en la subsección 2.1.2, merecen destacarse dos métodos que calculan los centros de las unidades ocultas mediante algoritmos supervisados:

- **Determinación de los centros por LVQ.** Asumiendo que la RNBR realizará tareas de clasificación o reconocimiento de patrones, y disponiendo del conjunto de entrenamiento con los vectores $\mathbf{x_i}$, etiquetados con la clase a la que pertenecen y_i, se puede utilizar el aprendizaje supervisado para determinar el conjunto de vectores prototipo $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. El algoritmo LVQ, [Kohonen, 1990], puede utilizarse para la determinación de estos vectores prototipo. Del algoritmo básico LVQ1 han derivado los algoritmos LVQ2, LVQ3 y OLVQ1. En [Kohonen, 1995] puede verse un tratamiento detallado de los algoritmos LVQ. Pues bien, los vectores prototipo pueden utilizarse como los centros de las neuronas ocultas de la RNBR [Schwenker et al., 1994].

- **Determinación de los centros mediante árboles de decisión.** Los árboles de decisión o de clasificación dividen el espacio de características $\mathbb{R}^n$ en regiones disjuntas $\Re_j$. De los posibles tipos de árboles de decisión el más utilizado es el árbol binario, donde cada nodo tiene 2 hijos o ninguno, y representa una cierta región $\Re$ del espacio $\mathbb{R}^n$. Si el nodo tiene 2 hijos, las regiones representadas por los nodos hijos, $\Re_{izquierda}$ y $\Re_{derecha}$, forman una partición de $\Re$:

$$\Re_{izquierda} \cup \Re_{derecha} = \Re$$

$$\Re_{izquierda} \cap \Re_{derecha} = \emptyset$$

  Los árboles de decisión utilizados habitualmente calculan cada partición con hiperrectángulos paralelos a los ejes del espacio de características.
  [Kubat, 1998] presenta un método para transformar dichas regiones hiperrectangulares disjuntas $\Re_j$ del espacio $\mathbb{R}^n$, en un conjunto de centros $\mathbf{c}_j \in \mathbb{R}^n$, y de desviaciones para inicializar una red de neuronas de base radial. [Schwenker and Dietrich, 2000] presenta varios algoritmos para inicializar los centros, las desviaciones y los pesos de las redes basándose en los árboles de clasificación.

2.1.3 Selección del modelo

El principal objetivo del entrenamiento de una red de neuronas no es encontrar un ajuste perfecto a los datos de entrenamiento, sino modelar el proceso

estadístico responsable de la generación de los datos, dado que lo realmente importante es la generalización o comportamiento de la red fuera de los datos de entrenamiento. El problema de la selección del modelo puede entenderse como un balance entre el "sesgo" (bias) y la varianza [Geman et al., 1992]. El error de generalización puede descomponerse en la suma del bias al cuadrado y la varianza, siendo el balance entre estos dos componentes un indicador de la complejidad del modelo. Un modelo muy simple tendrá un bias muy grande en el sentido de que el modelo, en término medio, diferirá bastante del deseado, incluso aunque se obtengan distintas instancias del modelo (cambiando los datos de entrenamiento, condiciones de inicialización, etc) que apenas diferirán unas de otras. Por otro lado, si el modelo es demasiado complejo, puede tener un bias bajo pero tendrá una gran varianza, y tenderá a ajustarse a los detalles del conjunto de entrenamiento.

La búsqueda del balance correcto entre bias y varianza se puede ver como el problema de encontrar el número correcto de parámetros libres del modelo. En el caso de RNBR esto implica la determinación del número óptimo de neuronas ocultas. Por este motivo, el análisis del efecto de añadir o quitar una neurona a la red es importante. Existen planteamientos que utilizan la fuerza bruta y consisten en entrenar varias redes con diferente número de neuronas ocultas y diferente número de patrones de entrenamiento, eligiendo la más apropiada utilizando validación cruzada o algún otro criterio [Moody, 1994]. Afortunadamente, debido a las características locales de las RNBR's y de los métodos de entrenamiento con fases separadas, pueden utilizarse alternativas diferentes a las de la fuerza bruta, existiendo dos líneas o enfoques principales: la primera de estas líneas utiliza la teoría de regularización comentada en la sección 2.1.1, y la segunda consiste en añadir o quitar neuronas ocultas a la red a medida que se está entrenando dando lugar a algoritmos de crecimiento o de poda de RNBR.

Regularización de Redes de Base Radial

La regularización es una técnica muy potente para favorecer ciertas soluciones frente a otras, añadiendo un funcional de penalización a la función de coste original, como ya se comentó en la sección 2.1.1. Se puede elegir un número adecuado de neuronas ocultas para una RNBR y entonces añadir un término de regularización a la función de coste para forzar la suavidad o alguna otra característica de la función. En [Friedman, 1994] se estudian varios tipos de penalizaciones que pueden ser aplicadas. Debido a la naturaleza lineal de la segunda etapa del entrenamiento de las RNBR's, se puede aplicar una forma de regularización sencilla y muy utilizada por los es-

tadísticos, llamada "ridge regression" o regresión contraída. En [Orr, 1995] y [Orr, 1996] puede verse una exposición detallada de este método que se comenta brevemente a continuación:

Regresión Contraída (*Ridge Regression*)

La regularización implica la utilización de un término de penalización en la función de coste para forzar a que la solución sea más sencilla. La regresión contraída no quita grados de libertad al modelo de forma explícita sino que hace algo equivalente reduciendo

la efectividad de los parámetros haciendo que el modelo tenga menos flexibilidad y, por tanto, sea menos sensible. Una forma de hacer esto en RNBR es añadir a la función de coste un término que penalice los pesos grandes:

$$Coste = \sum_{i=1}^{N} (t_i - y(\mathbf{x}_i))^2 + \lambda \sum_{j=1}^{m} w_j^2 \tag{2.55}$$

Siendo λ el parámetro de regularización que controla el balance entre ajustar los datos de entrenamiento y evitar la penalización por pesos grandes. Un valor pequeño de λ significa que el modelo puede ajustarse con precisión a los datos de entrenamiento sin causar una gran penalización, mientras que un valor grande de λ significa que el ajuste preciso a los datos de entrenamiento debe ser sacrificado si precisa de pesos grandes. El sesgo o bias introducido favorece soluciones con pesos pequeños y el efecto es suavizar la función de salida.

Regresión Contraída Local

Este método es una variación del anterior donde cada peso se penaliza de forma diferente, asociando un parámetro de regularización a cada función de base radial. La función de coste viene dada por:

$$Coste = \sum_{i=1}^{N} (t_i - y(\mathbf{x}_i))^2 + \sum_{j=1}^{m} \lambda_j w_j^2 \tag{2.56}$$

Aquí el efecto de cada parámetro de regularización es permitir que la suavidad de la función se adapte a las condiciones locales. Esto es especialmente útil cuando la función subyacente tiene diferentes propiedades de "rugosidad" o tiene complejidad diferente en diferentes partes del espacio de entrada.

Métodos de crecimiento y poda de Redes de Base Radial

Estos métodos adaptan la estructura de la red durante el entrenamiento. Los métodos de crecimiento o constructivos ([Fritzke, 1994], [Platt, 1991]) construyen de forma incremental la red, mientras que los métodos de poda ([Hassibi and Storkt, 1993], [Leonardis and Bischof, 1998]) comienzan con una red compleja y van eliminando unidades ocultas y/o pesos. A veces, los métodos de construcción y poda se combinan de forma iterativa [Yingwei et al., 1997]. A continuación se verán con más detalle algunos de los métodos citados.

Método constructivo de Platt (*The resource allocating network: RAN*)

Este método fue propuesto por Platt [Platt, 1991] y consiste en la incorporación de nuevas neuronas ocultas o en el reajuste de los parámetros de las neuronas existentes cada vez que se presenta un patrón. Si la respuesta de la red ante el nuevo patrón es deficiente, se asigna una nueva unidad para corregir la respuesta del patron presentado. Si la red se comporta bien ante un nuevo patrón, entonces los parámetros de las unidades existentes se actualizan utilizando el algoritmo estándar de mínimos cuadrados.

La red parte de un estado inicial con 0 unidades ocultas y se le comienzan a presentar patrones. Cuando se considera que un nuevo patrón $\mathbf{x}_t$ -presentado en el instante t- no está bien representado por la red, se asigna una nueva unidad centrada en dicho patrón, con una desviación proporcional a la distancia entre dicho patrón y el centro más cercano:

$$\begin{aligned} \mathbf{c}_n &= \mathbf{x}_t \\ \sigma_n &= \kappa \|\mathbf{x}_t - \mathbf{c}_{mas-cercano}\| \end{aligned} \tag{2.57}$$

Además, el peso de la conexión de esta neurona con la capa de salida se igualan a la diferencia entre la salida deseada para ese patrón $\mathbf{x}_t$ y la salida de la red y_t:

$$\mathbf{w}_n = \mathbf{t}_t - \mathbf{y}_t \tag{2.58}$$

Para considerar que un patrón presentado a la red no está bien representado por ella, deben cumplirse dos condiciones:

1. El nuevo patrón x_t está lejos de los centros actuales:

$$\|\mathbf{x}_t - \mathbf{c}_{mas-cercano}\| > \delta(t) \tag{2.59}$$

siendo $\delta(t)$ un valor de distancia o escala de resolución de la red en la presentación del patrón presentado en el instante t. Esta distancia comienza siendo máxima δ_{max} coincidiendo con el tamaño del espacio de entrada con densidad de probabilidad no nula, y va disminuyendo exponencialmente hasta que alcanza el valor mínimo δ_{min}, valor que coincide con la escala mínima que se considera de interés para la red.

2. La diferencia entre la salida deseada del nuevo patrón t_t y la salida de la red y_t es grande:

$$t_t - y_t > \epsilon \tag{2.60}$$

A continuación se presenta una descripción detallada del algoritmo:

$\delta = \delta_{max}$
iterar para todas las presentaciones de pares entrada/salida $(\mathbf{x}, \mathbf{t})$
{
evaluar la salida de la red $y_t = \sum_{j=1}^{m} w_j \phi_j(x_j)$
calcular el error de la red $E = t_t - y_t$
encontrar la distancia d al centro más cercano
si $E > \epsilon$ y $d > \delta$
{
asignar una nueva unidad oculta de centro $\mathbf{c}_{nuevo}$
establecer los pesos de esta nueva unidad con la capa de salida $w_{nuevo} = E$
si es la primera unidad que se asigna entonces
amplitud $\sigma_{nuevo} = \kappa\delta$
si no
amplitud $\sigma_{nuevo} = \kappa d$
}
si no
calcular por gradiente descendiente $\mathbf{c}_j, \mathbf{w}_j$
si $\delta > \delta_{min}$ entonces
$\delta = \delta \cdot exp(-1/\tau)$
}

El método de Platt fue mejorado utilizando un filtro de Kalman extendido, [Kadirkamanathan and Niranjan, 1993], para ajustar los parámetros de la red en lugar de utilizar el método de mínimos cuadrados. La red resultante fue llamada RANKEF (Resource Allocating Network via Extended Kalman Filter) y su rendimiento es superior en las tareas de aproximación de funciones y predicción de series temporales.

Método de poda de Leonardis y Bischof

El método de [Leonardis and Bischof, 1998] parte de una RNBR con una estructura sobredimensionada y se va reduciendo gradualmente su complejidad hasta llegar a una estructura más simple basándose en el método de Longitud de Descripción Mínima (MDL: Minimum Description Length) [Rissanen, 1984]. De acuerdo con el principio en que se basa el método MDL, de un conjunto inicial de RNBR se seleccionan aquellas que describen a los datos de entrenamiento con la codificación de longitud mínima. En lugar de entrenar muchas redes y luego seleccionar la mejor de acuerdo con el principio MDL, se efectúa la selección durante el entrenamiento consiguiendo menor coste computacional. Esta selección es la clave de este método, y es la que tiene la tarea de quitar las funciones radiales redundantes, de acuerdo con el principio MDL. A continuación se describirán los fundamentos de este método: Antes de modelar una función con una red, esta función solamente puede describirse especificando o enumerando las muestras. Después de construir la red, algunas de las muestras, o quizá todas, pueden ser descritas en términos de la red. Para representar la complejidad de la red, es decir, el número de funciones de base radial que posee, se utiliza un vector $\mathbf{m}^T = (m_1, m_2, \ldots, m_M)$ siendo M el número total de neuronas iniciales que posee la red. Los componentes del vector m_i indican la presencia de los correspondientes nodos, de forma que un 1 en la componente i indica la presencia de dicha neurona, y un 0 su ausencia. La longitud de codificación de una determinada función, $L_{funcion}$ modelada por una red puede darse como la suma de dos términos:

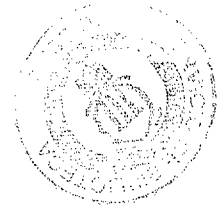
$$L_{funcion}(\mathbf{m}) = L_{muestras}(\mathbf{m}) + L_{red}(\mathbf{m}) \tag{2.61}$$

donde $L_{muestras}(\mathbf{m})$ es la longitud de codificación de las muestras individuales que no están descritas en términos de la red (la red no las representa) y $L_{red}(\mathbf{m})$ es la longitud de la codificación de las muestras representadas por la red. La idea principal consiste en seleccionar un subconjunto de nodos RBF que produzca la codificación de mínima longitud, es decir se debe maximizar la eficiencia de la descripción o codificación definida como:

$$E = 1 - \frac{L_{funcion}(\mathbf{m})}{L_{muestras}(\mathbf{0})} \tag{2.62}$$

donde $L_{muestras}(\mathbf{0})$ indica la longitud de la codificación de las muestras cuando la red no existe.

Para cada nodo i de una RNBR se pueden identificar los siguientes términos:



1. Un conjunto de muestras de entrada R_i que activan dicho nodo y que representan el dominio de dicho nodo. n_i será el cardinal de este conjunto.

2. El conjunto de parámetros del nodo: centro, anchura y pesos de las conexiones con las unidades de salida. N_i será el cardinal de este conjunto de parámetros.

3. La medida de ajuste del nodo, ξ_i que mide la diferencia entre los valores reales de la función y las aproximaciones producidas por la red, para el dominio R_i.

Teniendo en cuenta estos términos, la ecuación 2.61 se puede escribir:

$$L_{funcion}(\mathbf{m}) = K_1[n_{todos} - n(\mathbf{m})] + K_2\xi(\mathbf{m}) + K_3N(\mathbf{m}) \tag{2.63}$$

donde n_{todos} indica el número total de muestras del conjunto de entrenamiento y $n(\mathbf{m})$ en número de muestras representadas por la red. $N(\mathbf{m})$ indica el número de parámetros necesarios para describir la red y $\xi(\mathbf{m})$ indica el error producido por la red. K_1, K_2, K_3 son pesos que se pueden determinar teóricamente, en términos de la teoría de la información, o también pueden ajustarse para expresar las preferencias por alguna descripción en particular. En este punto, la tarea consiste en encontrar un vector $\hat{\mathbf{m}}$ tal que $L_{funcion}(\hat{\mathbf{m}})$ sea mínima, es decir:

$$L_{funcion}(\hat{\mathbf{m}}) = \min_m L_{funcion}(\mathbf{m}) \tag{2.64}$$

Teniendo en cuenta la ecuación 2.63, y dado que n_{todos} es constante, minimizar la ecuación equivale a maximizar la expresión:

$$F(\hat{\mathbf{m}}) = \max_m F(\mathbf{m}) = K_1n(\mathbf{m}) - K_2\xi(\mathbf{m}) + K_3N(\mathbf{m}) \tag{2.65}$$

Y este es el problema de optimización que tiene que resolver el método, para encontrar el máximo global de la función. En este tipo de problemas el número de soluciones se incrementa exponencialmente con el tamaño del problema, y no es razonable resolverlo de forma exhaustiva. Por tanto, se renuncia a encontrar la solución óptima para encontrar una solución "práctica". Teniendo en cuenta los fundamentos del método, la descripción básica del algoritmo es sencilla:

1. **Inicialización** Se inicializa la red asignando una función radial a cada muestra y estableciendo sus desviaciones en función de los centros más cercanos. Los pesos de la capa de salida se establecen utilizando un algoritmo regularizado.

2. **Adaptación** Se adaptan los centros, desviaciones y pesos con un algoritmo de entrenamiento que utilice gradiente descendiente.

3. **Selección** Se optimiza la función $F(\mathbf{m})$ respecto a m. Si $m_i = 0$, se elimina la función i de la red.

4. Si la selección del paso 3 no ha eliminado ninguna función y los cambios en los pesos son pequeños, termina el algoritmo. Si no, ir a paso 2.

Combinación de los métodos de crecimiento y poda

Como se comentó anteriormente, también existen métodos mixtos que combinan las técnicas de crecimiento con las de poda. Entre ellos destaca el algoritmo de [Yingwei et al., 1997] que combina el criterio de crecimiento que utiliza Platt en su trabajo ([Platt, 1991]) con una estrategia de poda basada en la contribución relativa de cada unidad oculta a la salida global de la red, obteniéndose como resultado una red de topología mínima. Los autores comparan el algoritmo propuesto con el de Platt, (RAN), y con [Kadirkamanathan and Niranjan, 1993], (RANKEF), para ciertos problemas de aproximación de funciones y predicción de series temporales, consiguiendo redes de menor número de neuronas que consiguen resultados iguales o mejores. Un inconveniente de los métodos RAN y RANKEF es que una vez que se crea una unidad oculta no puede eliminarse. Por este motivo, los métodos anteriores pueden producir unidades que inicialmente están activas, pero que pueden terminar contribuyendo poco a la salida de la red. La idea que ha motivado este nuevo método combinado es que si dichas neuronas pueden ser detectadas y eliminadas a medida que progresa el entrenamiento, se puede conseguir una red de topología más sencilla. Este método adopta la idea básica de RANKEF en cuanto a la estrategia de crecimiento e incorpora una estrategia de poda para obtener una red mínima. Al método resultante se le denomina M-RAM (Minimal RAN). A continuación se describe con más detalle la estrategia de poda del algoritmo, que tiene el objetivo de eliminar las unidades ocultas superfluas:
Sea ϕ_j la contribución de la neurona oculta j a la salida:

$$\phi_j = w_j \exp(-\frac{\|\mathbf{x} - c_j\|^2}{\sigma_j^2}) \tag{2.66}$$

Los valores de salida ϕ_j se deben examinar continuamente, para decidir si una unidad oculta debe eliminarse. Si la salida de una unidad es menor que un cierto valor umbral después de un número M de observaciones, esa unidad se elimina. Para evitar inconsistencias en la comparación de valores,

las salidas se normalizan respecto al valor de salida máximo. El algoritmo sería el siguiente:

- Para cada observación (x_i, y_i) calcular los valores de salida de todas las unidades ocultas ϕ_{ji} (salida de la unidad j para el patrón i), utilizando la ecuación 2.66.
- Encontrar el valor absoluto máximo $\|\phi_i^{max}\|$. Calcular los valores de las salidas normalizadas $r_{ji} = \|\frac{\phi_{ji}}{\phi_i^{max}}\|$.
- Eliminar las unidades cuyas salidas normalizadas ϕ_{ji} son menores que un valor umbral δ para M observaciones.
- Ajustar la dimensionalidad del Filtro de Kalman Extendido (utilizando en la estrategia de crecimiento) para que se ajuste a la red reducida.

2.1.4 Aplicaciones de las RNBR

Debido a que las RNBR's son aproximadores universales y disponen de algoritmos de entrenamiento rápidos, se han utilizado en una gran variedad de aplicaciones: predicción de series temporales caóticas, reconocimiento de lenguaje hablado, procesamiento de imágenes, diagnóstico médico, ecualización adaptativa en sistemas de comunicaciones, reconocimiento de lenguaje de manos, reconocimiento de caras ,etc. Todas las aplicaciones comentadas tienen que ver con un problema de mapeado subyacente y utilizan las capacidades de aproximación universal de las RNBR. Ya que muchas de estas aplicaciones utilizan la RNBR como un aproximador estático de funciones, puede surgir la duda de si sería más apropiado un perceptron multicapa para realizar esa tarea; la idoneidad de la elección depende de la naturaleza del problema. Para ciertas aplicaciones, si se eligen y se entrenan adecuadamente,los dos tipos de modelos obtienen resultados similares, aunque normalmente las RNBR son más rápidas de entrenar pero también son más sensibles a la *maldición de la dimensionalidad* [Friedman, 1994].

En [Ghosh and Nag, 2000] se muestra cómo se relacionan los clasificadores Bayesianos y las redes de base radial normalizadas, siendo estas redes muy efectivas en problemas de clasificación. Esta relación también se utiliza en aplicaciones tales como ecualizadores adaptativos [Mulgrew, 1996]: en trabajos iniciales, se comprobó que los ecualizadores basados en el perceptron multicapa eran superiores a los convencionales, pero por otra parte tenían graves inconvenientes entre los cuales se encontraban el gran tiempo de entrenamiento requerido y la carencia de una metodología para la selección de la arquitectura. Mulgrew comprobó que la estructura de las redes

de base radial y su estrecha relación con los métodos bayesianos para la ecualización de canales y rechazo de interferencias hacían de estas redes una solución muy buena para estos problemas.

Hay otras aplicaciones que se basan en relacionar las RNBR con los sistemas de inferencia borrosa [Jang and Sun, 1993] [Shim and Cheung, 1995]. En el primer trabajo se muestra que bajo ciertas restricciones menores el comportamiento funcional de las RNBR's y los sistemas de inferencia borrosa son realmente equivalentes. Esta equivalencia funcional implica que los avances en ambos campos, tales como nuevas reglas de aprendizaje, el análisis del poder de representación, etc., pueden aplicarse a ambos campos directamente. El gran interés de este trabajo radica en que dos modelos que proceden de campos tan diferentes resultan ser funcionalmente equivalentes. En estos trabajos, las funciones de base radial sirven para proporcionar funciones de pertenencia borrosas y sus salidas pasan a través de una o más capas de nodos para realizar la inferencia borrosa. Además, el algoritmo de retropropagación puede extenderse para modificar los centros de las funciones de base radial y sus anchuras y por tanto, las funciones de pertenencia borrosas [Shim and Cheung, 1995].

2.2 Métodos de aprendizaje perezoso

La mayoría de los métodos de aprendizaje supervisado pueden considerarse métodos de aprendizaje temprano, ya que la generalización se lleva a cabo más allá de los datos de entrenamiento, antes de observar la nueva muestra de test. Estos métodos construyen una descripción general y explícita de la función objetivo a partir de los ejemplos de entrenamiento. Cuando se recibe un patrón de test los métodos tempranos ya han elegido su aproximación global. Esa aproximación global sobre los datos de entrenamiento que representan el dominio puede conducir a pobres resultados en la generalización.

Un enfoque alternativo que trata de resolver el problema de la generalización, es retardar la fase de generalización hasta el momento en que obtengamos una nueva muestra de test, utilizando una selección de datos de entrenamiento en lugar de utilizar todo el conjunto disponible en el cual podría haber información irrelevante o redundante. Estos métodos se conocen en la literatura como métodos de aprendizaje retardado, perezoso o *lazy learning methods*, o también como métodos de aprendizaje basado en instancias (*Instance Based Learning methods* o IBL) [Mitchell, 1997], porque retrasan la decisión de cómo generalizar más allá de los datos de entrenamiento hasta el momento en que se recibe una nueva muestra. Estos métodos que derivan del clasificador de patrones por el vecino más próximo, [Cover and Hart, 1967], generalmente deben buscar o seleccionar datos relevantes que sirvan para responder a un nuevo patrón; es decir, la decisión de cómo generalizar se lleva a cabo cuando un patrón de test necesita ser respondido construyendo aproximaciones locales. La relevancia de cada patrón en el conjunto de datos de entrenamiento se determina habitualmente mediante una función de distancia, donde los puntos más cercanos tienen más relevancia. Una vez que los patrones relevantes se han seleccionado, las nuevas muestras se generalizan combinando los patrones más relevantes del conjunto de entrenamiento y descartando aquellos otros que no sólo no contribuyen a mejorar la generalización de la muestra, sino que podrían incluso empeorarla. En este sentido, se dice que los métodos perezosos construyen aproximaciones locales porque sólo se tienen en cuenta los datos locales en la generalización. Una ventaja esencial del aprendizaje retardado es que en lugar de estimar la función objetivo una sola vez para todo el espacio de instancias se estima localmente y de manera diferente para cada nueva instancia. Los métodos perezosos se diferencian unos de otros por sus funciones de predicción, por sus métricas de similitud y por las formas de seleccionar los ejemplos de entrenamiento. El valor de la nueva muestra se calcula considerando los valores del conjunto de instancias seleccionado.

Parece lógico pensar que utilizando técnicas de aprendizaje retardado puedan esperarse mejoras en la capacidad de generalización, porque solamente se utilizan patrones relevantes.

Según [Mitchell, 1997], los métodos de aprendizaje retardado se pueden dividir en tres grandes grupos:

- Algoritmo de K-vecinos.
- Algoritmo de K-vecinos ponderado por distancia.
- Regresión local ponderada.

A continuación se revisa cada uno de los apartados.

2.2.1 Algoritmo de k-vecinos

Es una de las técnicas más ampliamente utilizadas y tiene multitud de variantes. Este algoritmo, procede del clasificador de patrones de Cover y Hart [Cover and Hart, 1967] y se describe a continuación en su formulación más básica ([Mitchell, 1997]).

Se asume que todos los ejemplos corresponden a puntos en un espacio n-dimensional $\Re^n$. Los vecinos más próximos de una muestra se definen en términos de la distancia euclídea estándar. Sea x una instancia arbitraria; su representación vectorial en el espacio euclídeo de n dimensiones será:

$$\langle a_1(x), a_2(x), a_3(x)...a_n(x)\rangle$$

siendo $a_r(x)$ el valor del atributo r-ésimo de la instancia x. Con esta representación, la distancia euclídea entre dos instancias x_i y x_j es:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2}$$

La función objetivo puede ser discreta o real, es decir, puede utilizarse para problemas de clasificación o para aproximación de funciones.

1. **Funciones discretas**
   En los problemas de clasificación, las funciones objetivo son de la forma $f : \Re^n \rightarrow V$ donde V es un conjunto discreto. Cada ejemplo de entrenamiento es un par de la forma $\langle x, f(x)\rangle$ donde x es la instancia, representada por el vector de atributos, y f(x) el valor de la función

para esa instancia, es decir, la clase a la que pertenece esa instancia. Para una nueva muestra de test x_q que debe ser clasificada, el valor que devuelve el algoritmo $\hat{f}(x_q)$, que corresponde al valor estimado de $f(x_q)$, es el valor más común de la función f entre los k ejemplos de entrenamiento más próximos a x_q.

2. **Funciones continuas**
   En el caso de funciones objetivo continuas,$f : \Re^n \rightarrow \Re$, el algoritmo calculará el valor medio de los k vecinos más próximos en lugar de calcular el valor más común entre ellos:

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Trabajos realizados relacionados con K-vecinos. IB1, IB2, IB3

En [Aha et al., 1991] se presentan tres algoritmos de aprendizaje basado en instancias (IBL) denominados IB1, IB2 e IB3. Todos ellos se derivan del clasificador de Cover y Hart [Cover and Hart, 1967] y son muy similares a los algoritmos reducidos de vecinos más próximos como son [Hart, 1968], [Gates, 1972], [Dasarathy, 1980], que también almacenan y utilizan instancias seleccionadas para generar predicciones de clasificación. Algunos investigadores han mostrado que estos algoritmos reducidos pueden reducir los requisitos de almacenamiento de instancias con pequeñas pérdidas en la precisión de la clasificación, aunque no pudieron predecir el ahorro esperado en estos requisitos de almacenamiento. [Aha et al., 1991] consigue resultados en este aspecto: los requisitos de almacenamiento de instancias esperados son de orden polinomial respecto al tamaño de la frontera del concepto objetivo en el espacio de instancias. Los algoritmos reducidos no son incrementales y su principal objetivo es mantener perfecta consistencia con el conjunto de entrenamiento inicial. Estos algoritmos no toleran los problemas del mundo real como el ruido, con lo que se vuelven muy poco robustos. Los algoritmos de Aha son incrementales y uno de sus objetivos es maximizar la precisión en la clasificación a medida que se van presentando nuevas instancias. Algunos de los principales problemas de los algoritmos IBL como los excesivos requisitos de almacenamiento y la intolerancia al ruido, se resuelven en estos algoritmos incrementales.

Metodología y marco de trabajo de los algoritmos IB1, IB2, IB3

Se asume que cada instancia está representada por el mismo número de atributos, y que es posible que algún atributo no tenga valor. Este

conjunto de atributos define un espacio n-dimensional denominado espacio de instancias. Uno de estos atributos corresponde a la categoría o clase a la que pertenece la instancia. La única entrada que tendrán estos algoritmos será una secuencia de instancias. La salida principal de los algoritmos es una *Descripción del Concepto* o *Concepto*, que es una función que mapea instancias en categorías, es decir, dada una instancia tomada del espacio de instancias, la función produce una clasificación que es el valor predicho para el atributo de categoría de esta instancia.

Un *Concepto* basado en instancias incluye un conjunto de instancias almacenadas y, posiblemente, alguna información referente al resultado de clasificaciones anteriores como, por ejemplo, el número de predicciones correctas o incorrectas. Este conjunto de instancias puede cambiar después de procesar cada instancia de entrenamiento. Los tres componentes fundamentales de estos algoritmos son:

1. *Función de similitud.* Esta función mide numéricamente la similitud entre una instancia de entrenamiento i y las instancias del Concepto.

2. *Función de clasificación.* Recibe los resultados de la función de similitud y los registros de funcionamiento de las clasificaciones y produce una clasificación para la instancia i.

3. *Actualizador de la Descripción del Concepto.* Mantiene los registros del funcionamiento de la clasificación y decide qué instancia incluir en la Descripción del Concepto. La entrada incluye la instancia i, los resultados de similitud, los resultados de clasificación y la Descripción del Concepto actual. Produce como salida la Descripción del Concepto modificada.

Los algoritmos IBL asumen que las instancias similares producen clasificaciones similares. Por esto, clasifican a las nuevas instancias de acuerdo con la clasificación del vecino más similar. También asumen estos algoritmos que todos los atributos tendrán la misma relevancia, por lo que se deberán normalizar los rangos de los posibles valores de los atributos.

Algoritmo IB1

Es el algoritmo más simple. La función de similitud que utiliza es la siguiente:

$$Similitud(x, y) = -\sqrt{\sum_{i=1}^{n} f(x_i, y_i)} \tag{2.67}$$

donde n es el número de atributos. Se define:

$$f(x_i, y_i) = (x_i - y_i)^2 \tag{2.68}$$

para los atributos numéricos, y

$$f(x_i, y_i) = (x_i \neq y_i) \tag{2.69}$$

para los atributos booleanos o simbólicos. Si falta un atributo, se considera diferente del atributo de la instancia con la que se compara. Si los dos faltan, entonces $f(x_i, y_i) = 1$.
El algoritmo es el siguiente:

Descripcion del Concepto(DC) $\leftarrow \emptyset$
para cada instancia $x \in$ Conjunto de entrenamiento hacer
    para cada $y \in DC$ hacer
      Simil[y] $\leftarrow$ Similitud(x, y)
    $y_{max} \leftarrow$ algún y $\in DC$ tal que Simil[y] sea maxima
    Si clasif (x) = clasif (y_{max}) entonces
      clasificacion $\leftarrow$ correcta
    si no
      clasificacion $\leftarrow$ incorrecta
    $DC \leftarrow DC \cup \{x\}$

[Aha et al., 1991] muestra que IB1 se puede aplicar a una amplia clase de conceptos y que el límite superior del número de instancias que necesita almacenar para aprender un concepto es de orden polinomial respecto al tamaño de la frontera del concepto. Los autores comparan IB1 con regresión lineal en tareas de predicción de funciones y se concluye que IB1 tiene algunas ventajas sobre la regresión: Consigue mayor precisión, el algoritmo de entrenamiento es más simple, no requiere un modelo de la función objetivo y funciona bien con atributos simbólicos. Un gran inconveniente de IB1 es que almacena todas las instancias de entrenamiento y cada predicción de una nueva instancia supone medir la distancia entre la nueva instancia y todas las almacenadas, por lo que se convierte en muy ineficiente cuando el conjunto de entrenamiento es grande. En [Aha et al., 1991] se mostró que los requisitos de almacenamiento de IB1 pueden ser reducidos significativamente sin perder apenas precisión en la clasificación y se desarrolló un algoritmo de entrenamiento perezoso con almacenamiento reducido llamado IB2.

Algoritmo IB2

En [Aha et al., 1991] se observó que sólo las instancias cerca de las fronteras de las clases eran útiles. Por tanto, se puede conseguir una gran reducción en los requisitos de almacenamiento almacenando sólo las instancias que aportan información. Desafortunadamente, este conjunto de instancias "infomativas" no se puede conocer sin un conocimiento completo de las fronteras del concepto, aunque, sin embargo, se puede aproximar con el conjunto de instancias mal clasificadas. Esta es la base del algoritmo IB2, que es idéntico a IB1 excepto en que solamente se almacenan las instancias mal clasificadas.

Los resultados con IB2 son casi tan buenos como los obtenidos con IB1, aunque con mucho menor número de instancias almacenadas, sobre todo en las aplicaciones donde la distancia de las instancias a la frontera del concepto varía mucho. Como puede suponerse, la mayoría de las instancias almacenadas se encuentran cerca de las fronteras de las clases. El ahorro en espacio de almacenamiento es mucho mayor cuando ninguna de las instancias tiene ruido. Además, en [Aha et al., 1991] se comprueba que IB2 es más sensible al ruido en el sentido de que a medida que éste aumenta, la precisión en la clasificación disminuye más rápidamente en IB2 que en IB1. Esto se explica porque las instancias de entrenamiento con ruido casi siempre están mal clasificadas, y como IB2 sólo almacena los ejemplos mal clasificados, la mayor parte de ellos serán ejemplos con ruido, con los cuales se generarán las decisiones de clasificación. En general, las instancias sin ruido consiguen altos grados de precisión en la clasificación, con un suficiente número de intentos; sin embargo, las instancias con ruido siempre obtendrán bajos rendimientos en la clasificación.

Algoritmo IB3

Dado que IB2 reduce significativamente los requisitos de almacenamiento de IB1 pero es muy sensible al ruido, en [Aha et al., 1991] se modifica el algoritmo para disminuir esta sensibilidad dando lugar al denominado algoritmo IB3 que emplea un filtro sencillo [Markovich and Scott, 1989] para determinar cuáles de las instancias almacenadas deben ser utilizadas para generar decisiones de clasificación.
La función de similitud es la misma que en IB1 e IB2, pero se modifican la función de clasificación y el algoritmo de actualización de la siguiente forma:

- IB3 dispone de un registro de clasificación para cada instancia almacenada, donde se anota el número de intentos de clasificación correcta e incorrecta. De este modo, se dispone de la historia del comporta-

miento de cada instancia y se puede prever cómo se comportará en el futuro.

- IB3 realiza un test con las instancias para decidir cuáles son buenas clasificadoras y cuáles se supone que tienen ruido. Las primeras se utilizarán para clasificar a las instancias que se presenten después y las segundas se eliminarán de la descripción del concepto.

Al analizar el comportamiento de IB3 en un dominio simple, se obtiene como conclusión que supera a IB2 tanto en términos de ahorro de espacio de almacenamiento como en precisión en la clasificación, ya que es capaz de filtrar el ruido. Se observa que cuando los niveles de ruido son muy altos, la cantidad de instancias almacenadas se aproxima a 0 puesto que muy pocas instancias obtienen buenos resultados de clasificación cuando el nivel de ruido es alto. Cuando el nivel de ruido es inferior al 45%, IB3 es capaz de detectar y eliminar casi todas las instancias con ruido. Debido a que este algoritmo utiliza muy pocas instancias ruidosas para clasificar las instancias de entrenamiento presentadas a continuación, su capacidad de clasificación se degrada más lentamente que la de IB1 e IB2. En [Aha et al., 1991] también se compara el comportamiento de IB3 con IB1 e IB2 en dominios reales, y se observa que mejora claramente los resultados de IB2 e IB1 tanto en términos de precisión en la clasificación como en ahorro de espacio de almacenamiento.

2.2.2 Algoritmo de k vecinos ponderado por distancia

Una opción para refinar este algoritmo consiste en asignar un peso a cada vecino seleccionado, en función de su distancia a la muestra de test. Es común el uso de la función inversa de la distancia para calcular el peso. De esta forma, el peso w_i del ejemplo x_i será:

$$w_i = \frac{1}{d(x_q, x_i)} \tag{2.70}$$

En el caso de funciones objetivo continuas, el valor que devuelve el algoritmo será:

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \tag{2.71}$$

donde w_i es el peso que se asigna a cada instancia y está definido en la ecuación 2.70. El denominador en la ecuación 2.71 es una constante de normalización.

Las variantes de k-vecinos comentadas anteriormente solamente consideran los k vecinos más próximos a la nueva muestra, pero si se ponderan los vecinos en función de la distancia, no hay ningún problema en considerar todas las muestras del espacio de entrada, ya que los más lejanos tendrán muy poca influencia en la estimación de $\hat{f}(x_q)$. El único inconveniente de tomar todos los ejemplos es el coste computacional. Si se toman en consideración todos los ejemplos, el método será *global*; si sólo se consideran los vecinos más próximos, el método será *local.*

[C.G.Atkenson et al., 1997] hacen una revisión completa de los métodos locales ponderados, dando especial relevancia a las funciones de similitud o funciones de distancia y a las funciones kernel o funciones de ponderación.

2.2.3 Regresión local ponderada

Es otra forma de aprendizaje perezoso y supone una generalización del método descrito en el párrafo anterior: en k-vecinos se hace una aproximación de la función objetivo para el punto concreto $x = x_q$, mientras que la regresión local ponderada construye una aproximación explícita a la función objetivo sobre una región local que rodea a x_q, [C.G.Atkenson et al., 1997], [Mitchell, 1997],[Vapnik, 1992]. Para construir esta función local se utilizan los ejemplos de entrenamiento de la vecindad de x_q. Esta técnica se denomina *regresión* porque este es el término ampliamente utilizado en estadística para el problema de aproximación de funciones reales, *local* porque la función es aproximada utilizando solamente datos cercanos a la nueva muestra y *ponderada* porque la contribución de cada ejemplo de entrenamiento depende de su distancia a la nueva muestra.

Dada una nueva instancia x_q, se debe construir una aproximación $\widehat{f}$ que se adapte a los ejemplos de entrenamiento en la vecindad de x_q. Esta función $\widehat{f}$ se usa para calcular el valor $\widehat{f}(x_q)$ que es la salida estimada para la nueva muestra. Esta descripción de $\widehat{f}$ puede entonces eliminarse, porque para cada nueva muestra debe calcularse una aproximación local distinta.

Consideremos el caso donde la función objetivo f es aproximada en la vecindad de x_q utilizando una función lineal de la forma:

$$\widehat{f}(x) = w_0 + w_1 a_1(x) + ...w_n a_n(x)$$

siendo $a_i(x)$ el valor del atributo i de la instancia x. Ahora hay que determinar los valores de los parámetros w_i de forma que se minimice el error cuadrático sobre patrones de entrenamiento. Según este planteamiento, se pueden utilizar tres criterios:

1. Minimizar el error sólamente sobre los k vecinos más próximos.

$$E_1(x_q) = \frac{1}{2} \sum_{x \in conj\ k\ vecinos\ de\ x_q} (f(x) - \widehat{f}(x))^2$$

2. Minimizar el error sobre el conjunto entero de entrenamiento (D) ponderando el error de cada ejemplo con una función (K) decreciente de la distancia a la muestra x_q.

$$E_2(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \widehat{f}(x))^2 K(d(x_q, x))$$

3. Utilizar una combinación de los apartados 1 y 2, es decir, solamente se utilizarán los k vecinos más próximos y además ponderando el error de cada ejemplo con una función decreciente con la distancia a la muestra x_q

$$E_3(x_q) = \frac{1}{2} \sum_{x \in conj\ k\ vecinos\ de\ x_q} (f(x) - \widehat{f}(x))^2 K(d(x_q, x))$$

Normalmente se elije el criterio 3, donde el coste computacional depende sólo del valor k y no del tamaño del conjunto original de entrenamiento. Con este criterio, utilizando la regla del gradiente descendiente se obtiene la siguiente regla para la determinación de los parámetros w_i :

$$\Delta w_j = \eta \sum_{x \in conj\ k\ vecinos\ de\ x_q} (f(x) - \widehat{f}(x)) a_j(x) K(d(x_q, x))$$

Existen en la literatura muchos métodos para ponderar los ejemplos de entrenamiento en función de la distancia, así como métodos para aproximar la función objetivo. Aunque se podrían utilizar funciones cuadráticas u otras más complejas, se suelen utilizar funciones lineales porque el coste computacional es menor y además aproximan con una precisión aceptable en un área suficientemente pequeña del espacio de entrada.

2.3 Discusión

En este capítulo se han revisado las RNBR y los métodos de aprendizaje retardado o de *lazy learning*. A continuación se resumen las ventajas e inconvenientes de ambos modelos:

Ventajas e inconvenientes de las RNBR

- **Deficiente capacidad de generalización de las RNBR.** Las RNBR son modelos robustos, tolerantes al ruido y pueden aproximar, con el grado de precisión deseado, cualquier relación no lineal entre la entrada y la salida, pero tienen dificultades para generalizar adecuadamente. En general, para poder construir una aproximación mediante la suma de aproximaciones locales, se requiere un número grande de unidades ocultas, especialmente si el número de dimensiones del espacio de entrada es grande, lo cual puede influir negativamente en la capacidad de generalización [Haykin, 1999]. Son necesarias métodos sofisticados, entre los que se encuentran los métodos de regularización y regresión contraída, para que aumente su precisión en la generalización.

- **Selección del modelo crítica.** La capacidad de generalización es sensible al modelo seleccionado adecuado para un conjunto de entrenamiento dado. Existe algoritmos sofisticados que tratan este problema, determinando la arquitectura óptima para un dominio determinado. Entre estos métodos destacan el método de Platt y el de Bischof.

- **Entrenamiento muy rápido.** El entrenamiento de las RNBR requiera un coste computacional mucho menor que otros modelos de redes de neuronas. Esto se debe a la característica local de este tipo de redes, de forma que cuando se presenta un patrón de entrenamiento, sólo se activa una neurona (o un grupo muy reducido de neuronas); por ello, sólo hay que ajustar el peso de esa neurona (o de ese grupo reducido de neuronas), y esta operación requiere un bajo coste computacional. Además, el número de ciclos de entrenamiento necesarios para ajustar los pesos es, en general, muy pequeño comparado con otros modelos. Esto se debe a la naturaleza lineal de la capa de salida.

Ventajas e inconvenientes de los métodos de aprendizaje retardado

- **Buena capacidad de generalización.** Al construir representaciones locales de la función objetivo para cada nueva muestra de test,

basándose solamente en los patrones más relevantes, la precisión en la capacidad de generalización es, en general, alta.

- **Problemas con la función de distancia en k vecinos.** La calidad de la generalización de k vecinos depende mucho de qué instancias son consideradas más cercanas a la muestra a generalizar, lo cual está determinado por la función de distancia utilizada. Muchas veces esta función de distancia permite que atributos redundantes, irrelevantes, o con ruido tengan tanto efecto en el cómputo de la distancia como los atributos más importantes. Puede darse el caso de instancias donde los atributos relevantes son casi idénticos y sin embargo están muy distantes entre sí porque difieren bastante en atributos poco importantes. Cuando ocurre esto, los métodos no consiguen generalizar adecuadamente. Esto ha hecho que aparezcan muchas variantes de k vecinos que utilizan atributos ponderados para calcular la distancia. En [Wettschereck et al., 1997] se comparan y analizan diferentes métodos que utilizan atributos con pesos. Se resalta un grupo de algoritmos donde los pesos de los atributos se van adaptando con el objetivo de lograr una buena generalización. Hay otras alternativas que consisten en eliminar completamente los atributos irrelevantes. En [Moore and Lee, 1994] se discuten métodos de validación cruzada para seleccionar subconjuntos de atributos relevantes del conjunto total de atributos para los algoritmos de k vecinos.

- **El problema de la elección del parámetro k.** La capacidad de generalización de los métodos anteriores depende de la correcta elección del parámetro k. Algunos investigadores [Duda and Hart, 1973] y [Aha, 1973] han mostrado que la precisión en la generalización de k vecinos aumenta con k, alcanza un máximo y luego disminuye a medida que k crece. El valor óptimo de k varía según los problemas, por tanto es muy importante determinar el mejor k para cada problema. En los primeros trabajos, el propio usuario determinaba el parámetro k. En trabajos posteriores, se determina automáticamente el valor óptimo de k utilizando validación cruzada, *leave-one-out cross-validation*. En este método, se predice el valor de cada instancia en el conjunto de entrenamiento, utilizando las instancias seleccionadas del resto del conjunto para diferentes valores de k, siendo el k óptimo el que produce el menor error medio sobre todas las instancias del conjunto de entrenamiento. El mejor k, por tanto, es un valor fijo para un determinado problema. En [Zhang et al., 1997] se muestra que el

mejor valor de k depende de dónde esté localizada la nueva muestra y del número de instancias disponibles en esa zona.

- **Alto coste computacional.** Los métodos de aprendizaje retardado, por su propia naturaleza, tienen un alto coste computacional, debido a que para cada muestra de test hay que construir una representación local de la función objetivo.

Capítulo 3

Objetivos de la tesis doctoral

Como se ha visto en el capítulo 2, las **redes de neuronas de base radial** son aproximadores universales, pudiendo aproximar cualquier relación no lineal entre la entrada y la salida. Son modelos robustos y, debido al carácter local de este tipo de redes y al carácter lineal de su capa de salida, su entrenamiento es muy rápido en comparación con otros modelos de redes de neuronas. Las RNBR necesitan un gran número de neuronas ocultas, especialmente si la dimensionalidad del espacio de entrada es grande, para realizar buenos ajustes a las funciones objetivo, y esto afecta negativamente a su capacidad de generalización, siendo éste un inconveniente de este tipo de redes.

Por otra parte, los **métodos de aprendizaje retardado** construyen las representaciones de la función objetivo de forma local dependiendo de la nueva muestra de test. Estos métodos tienen, en general, una gran capacidad de generalización. Pero la precisión en la generalización de los métodos de aprendizaje retardado depende en gran medida del número de patrones que se seleccionen, siendo éste valor un parámetro crucial que hay que determinar. Además, en estos métodos la calidad de la generalización también depende mucho de qué instancias son consideradas más cercanas a la muestra a generalizar, lo cual está determinado por la función de distancia utilizada. Puede darse el caso de instancias donde los atributos relevantes son muy parecidos y sin embargo están muy distantes entre sí porque difieren bastante en atributos poco importantes. Cuando ocurre esto, los métodos de aprendizaje retardado no consiguen generalizar adecuadamente.

3.1 Objetivos

El objetivo principal de esta tesis doctoral consiste en mejorar la capacidad de generalización de las redes de neuronas de base radial, utilizando para ello métodos de entrenamiento basados en el aprendizaje retardado. Se pretende aprovechar las ventajas de ambos modelos de forma que, para cada nueva muestra de test, se seleccionen de forma automática los patrones de entrenamiento más adecuados para entrenar la red, descartando así los ejemplos que no sólo no proporcionan información útil a la red sino que además podrían distorsionar el proceso de aprendizaje. Se seguirá el principio de que cuanto más cerca esté, en términos de distancia euclídea, el patrón de entrenamiento de la nueva muestra de test, más importancia deberá tener en el entrenamiento de la red.

Otro objetivo de esta tesis es que el método propuesto sea de aplicación general, aplicable a cualquier modelo de red de neuronas. Aunque el método se propone, en principio, para mejorar la capacidad de generalización de las RNBR, ya que este tipo de red resulta muy adecuado dada su gran rapidez en el entrenamiento, se pretende que sea aplicable independientemente del modelo de red de neuronas elegido.

Además, se pretende que no sea necesario determinar el número de patrones de entrenamiento necesarios para un determinado problema. Según sea el patrón de test, se podrá necesitar un número diferente de patrones de entrenamiento, y éstos deberán ser seleccionados automáticamente.

De entre los patrones de entrenamiento seleccionados, como se comentó anteriormente, los más similares al patrón de test deberán tener más importancia en el entrenamiento. Esta similitud estará relacionada con la distancia euclídea, de forma que cuanto más cerca esté un patrón de entrenamiento del patrón de test, más similar a él será. Esta distinta importancia de cada patrón en el aprendizaje de la red se conseguirá replicando los patrones de entrenamiento, de forma que los más similares al patrón de test sean repetidos más veces.

El coste computacional, aunque mayor que el correspondiente al entrenamiento convencional, no debe ser demasiado alto. Debe aprovecharse la gran rapidez de las RNBR en el entrenamiento.

El método debe ser robusto, teniendo poca dependencia de parámetros y de inicializaciones aleatorias.

3.2 Evaluación de la Tesis Doctoral

Para validar la consecución de los objetivos, se realizarán experimentos sobre varios dominios de diferente naturaleza: aproximación de funciones, series temporales y clasificación. En todos ellos se medirá el error medio sobre todos los patrones del conjunto de test en función del número de neuronas de las redes y de otros parámetros que dependerán de las diferentes propuestas del método. Estos parámetros serán la desviación o anchura de la función kernel en el caso de la ponderación gaussiana, y el radio relativo de la hiperesfera de selección, en el caso de la ponderación inversa. En algunos casos, además de medir el error medio para todos los patrones del conjunto de test, se medirá el error producido para cada patrón de test, con el objeto de representarlo gráficamente, pues esto da una idea más completa del comportamiento del método. En el método propuesto los centros de las RN-BR son situados utilizando el algoritmo K-medias, que en su versión clásica realiza una inicialización aleatoria de dichos centros. Con el fin de estudiar la influencia de las inicializaciones aleatorias en el comportamiento de las redes, se evalúan los errores de generalización para un determinado patrón en función del radio relativo de la hiperesfera de selección y para distintas inicializaciones aleatorias.

Capítulo 4

Selección de patrones de entrenamiento mediante vecindad ponderada

El principal objetivo de esta tesis es mejorar la capacidad de generalización de las redes de neuronas de base radial (RNBR), actuando sobre el conjunto de entrenamiento, con un planteamiento basado en aprendizaje retardado. De esta forma, cada vez que se reciba una muestra de test se seleccionarán, del conjunto de entrenamiento, los patrones más similares al patrón de test y con ellos se entrenará la red para que pueda responder a dicho patrón de test. Además, cuanto más cerca esté el patrón de entrenamiento del patrón de test, más importancia tendrá en el entrenamiento. Para conseguir este objetivo se propone un método de selección de patrones de entrenamiento mediante vecindad ponderada, es decir, se seleccionarán como patrones de entrenamiento los correspondientes a la vecindad del patrón de test, y además se ponderarán de forma inversamente proporcional a la distancia a este patrón de test, de modo que los patrones más cercanos a la muestra de test tendrán más importancia. A lo largo del capítulo también se hará referencia a este método con el nombre de *método selectivo*.

En este capítulo se describen diferentes propuestas del método de selección de patrones de entrenamiento mediante vecindad ponderada. Estas propuestas se diferencian en la función utilizada para seleccionar los patrones de entrenamiento, así como en diferentes modos de entrenar las RNBR, como se explicará en las secciones correspondientes.

Para validar el método, se han utilizado dos dominios de aproximación de funciones, dos dominios de predicción de series temporales y un dominio de

clasificación. Se han entrenado las RNBR por el método convencional para medir su capacidad de generalización en los dominios anteriores y así poder comparar estos resultados con los obtenidos por las diferentes propuestas del método de aprendizaje selectivo.

El resto del capítulo está organizado del siguiente modo: en la sección 4.1 se describen los dominios utilizados en la experimentación. En la sección 4.2 se muestran los resultados obtenidos para los dominios descritos cuando se entrenan las redes de forma convencional, para que puedan compararse con los resultados obtenidos cuando se utilizan las diferentes variantes del método selectivo propuesto en esta tesis. En la sección 4.3 se describe la primera propuesta del método selectivo denominada *ponderación gaussiana* para seleccionar los patrones. Se ha denominado así a dicha propuesta porque se ha utilizado la función gaussiana como función kernel para seleccionar a los patrones de entrenamiento. En la sección 4.4 se describe la segunda propuesta, donde se utiliza la denominada *ponderación inversa* por utilizar como función kernel dicha función. Esta sección está dividida en 4 subsecciones con diferentes variantes del método de ponderación inversa.

4.1 Descripción de los dominios

- Dominios de aproximación
  - Función Definida por Partes
  - Polinomio de Hermite
- Dominios de predicción de series temporales
  - Serie temporal de las mareas de Venecia
  - Serie temporal de Mackey-glass
- Dominio de clasificación
  - Dominio Pima-Indians Diabetes

4.1.1 Dominios de Aproximación

Función Definida por Partes

La función definida por partes es una función de una variable definida en tres intervalos del eje x, según la ecuación 4.1. Esta función se ha elegido debido a los pobres resultados que presentan las redes de base radial cuando la aproximan.

$$f(x) = \begin{cases} -2.186x - 12.864 & si \quad 10 \leq x < -2 \\ 4.246x & si \quad -2 \leq x < 0 \\ 10e^{-0.05x-0.5}\sin(0.03x + 0.7x) & si \quad 0 \leq x \leq 10 \end{cases} \tag{4.1}$$

El *conjunto de entrenamiento* está compuesto por 120 patrones generados aleatoriamente por una distribución uniforme en el intervalo $[-10, 10]$. El *conjunto de test* está compuesto por 80 puntos generados de la misma forma. Los datos se normalizan en el intervalo $[0, 1]$. En la figura 4.1 se muestra la representación gráfica de dicha función con los datos normalizados.

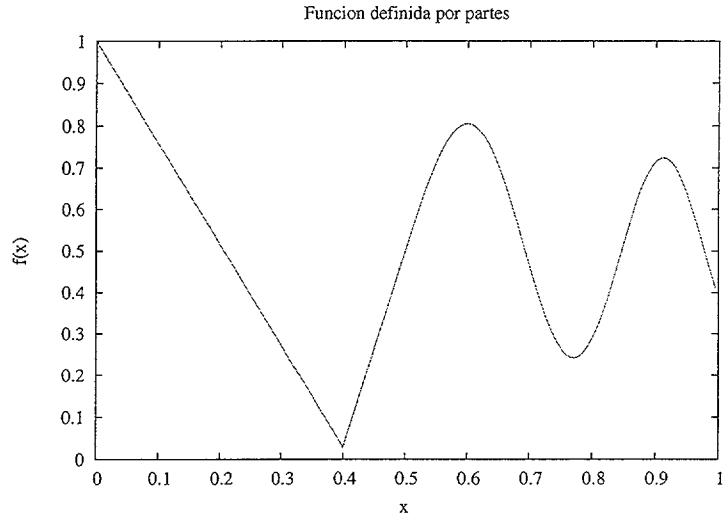


Figura 4.1:
Función Definida por Partes

Polinomio de Hermite

Esta función es ampliamente utilizada en la literatura, para estudiar la capacidad de generalización de las redes de neuronas en general, y especialmente en las RNBR [Orr, 1995], [Leonardis and Bischof, 1998] ,[Yingwei et al., 1997], [Kadirkamanathan and Niranjan, 1993].
El polinomio de Hermite viene dado por la siguiente ecuación:

$$f(x) = 1.1(1 - x + 2x^2)e^{-\frac{1}{2}x^2} \tag{4.2}$$

Los conjuntos de entrenamiento y test se han generado, al igual que se ha hecho en los artículos citados, de la siguiente forma: El *conjunto*

de entrenamiento está formado por 40 muestras generadas aleatoriamente según una distribución uniforme en el intervalo $[-4, 4]$. El *conjunto de test* está formado por 200 muestras generadas del mismo modo. Los datos están normalizados en el intervalo $[0, 1]$.

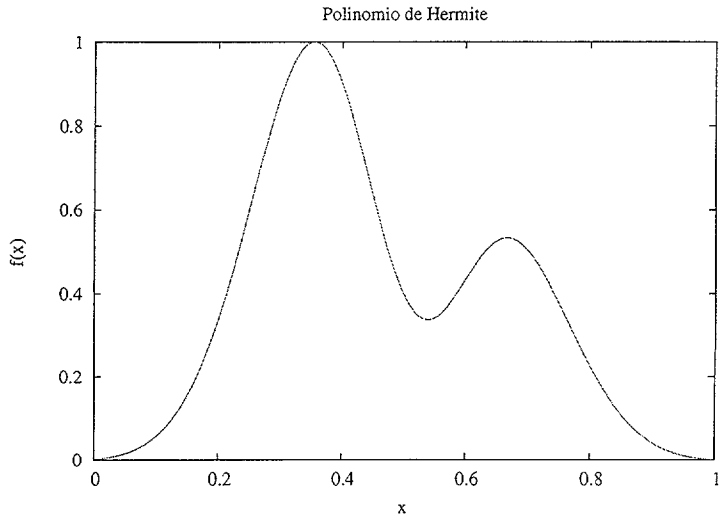


Figura 4.2:
Polinomio de Hermite

En la figura 4.2 se muestra la representación gráfica de la función, con los datos normalizados

4.1.2 Dominios de Predicción de Series Temporales

Serie temporal de las Mareas de Venecia

En este dominio se representa un problema natural que consiste en la predicción del nivel del agua en la laguna de Venecia.
Las mareas inusualmente altas se producen por la combinación de factores climáticos que tienen un comportamiento caótico, con otros factores periódicos asociados con un área particular y que podrían resultar fácilmente modelables. La predicción de estos eventos siempre ha sido un tema de gran interés, tanto desde el punto de vista humano como económico. El nivel del agua en la laguna de Venecia es un claro ejemplo de estos eventos [Michelato et al., 1983]. El ejemplo más famoso de inundaciones en la laguna de Venecia ocurrió en noviembre de 1966 cuando el nivel del agua superó

en casi 2 metros el nivel normal. Este fenómeno es conocido por el término de "agua alta" y se han hecho muchos esfuerzos en Italia para desarrollar sistemas de predicción del nivel del agua en Venecia, y especialmente del fenómeno del "agua alta" [Tomasin, 1973]. Aunque se ha afrontado el problema de la predicción del nivel del agua en Venecia de diversas maneras, recientemente se han utilizado Redes de Neuronas Multicapa con Conexiones hacia Adelante, [Zaldívar et al., 2000], obteniendo ciertas mejoras sobre los modelos tradicionales.

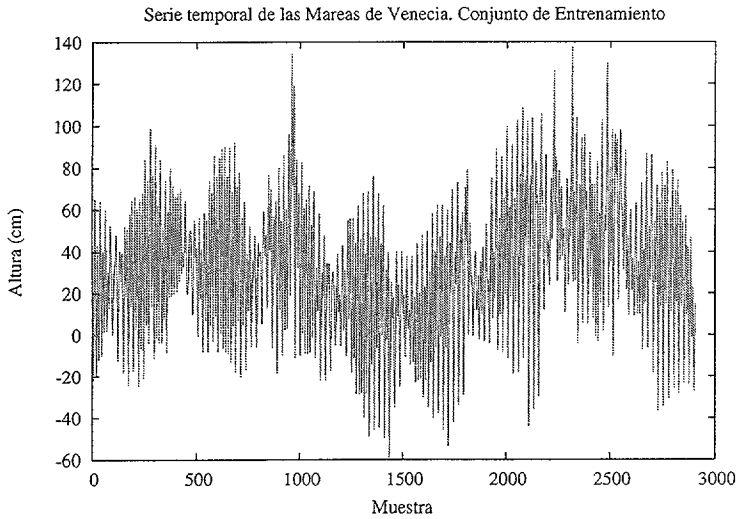


Figura 4.3:
Serie temporal de las Mareas de Venecia. Conjunto de Entrenamiento

Hay una gran cantidad de datos disponibles representando el comportamiento de la serie temporal. Sin embargo, los datos correspondientes al comportamiento estable del agua son muy abundantes frente a los que corresponden a las situaciones de "agua alta". Esta situación lleva a la siguiente conclusión. una red entrenada con el conjunto de entrenamiento completo no será muy precisa para predecir las situaciones de "agua alta". Este dominio es muy apropiado para ser utilizado con aprendizaje selectivo pues parece natural que si la red se entrena con los patrones seleccionados, mejorará sus predicciones. Se dispone de una muestra de datos, que corresponden al nivel del agua en la laguna de Venecia entre 1980 y 1994 medido cada hora. El objetivo consiste en predecir el siguiente valor de la serie temporal, mediante un modelo no lineal que utiliza los valores de la serie en los n momentos

anteriores. En nuestro caso, hemos elegido el valor $n = 6$. Cuando se haga una predicción a más largo plazo, serán más adecuados los modelos a los que se les aporte mayor información.

El *conjunto de entrenamiento* del que disponemos para realizar los experimentos corresponde a un periodo de tiempo de 2996 horas de duración; por tanto dicho conjunto contiene 2996 puntos, algunos de los cuales representan situaciones de marea alta. En la figura 4.3 se muestra la representación de los datos correspondientes al conjunto de entrenamiento. En ella se ve representada la altura del agua medida en centímetros en función del tiempo medido en horas.

Como puede observarse, entre las casi 3000 muestras, se dan muy pocos casos de marea alta. El *conjunto de test* corresponde a un periodo de tiempo posterior al del conjunto de entrenamiento, compuesto por 20 puntos donde se da una situación de marea alta, que es la que interesa predecir. En la figura 4.4 se muestra la representación gráfica de dichos puntos en función del tiempo.

Aunque en las gráficas se han representado los datos originales, se han normalizado en el intervalo [0,1] como en el resto de los dominios.

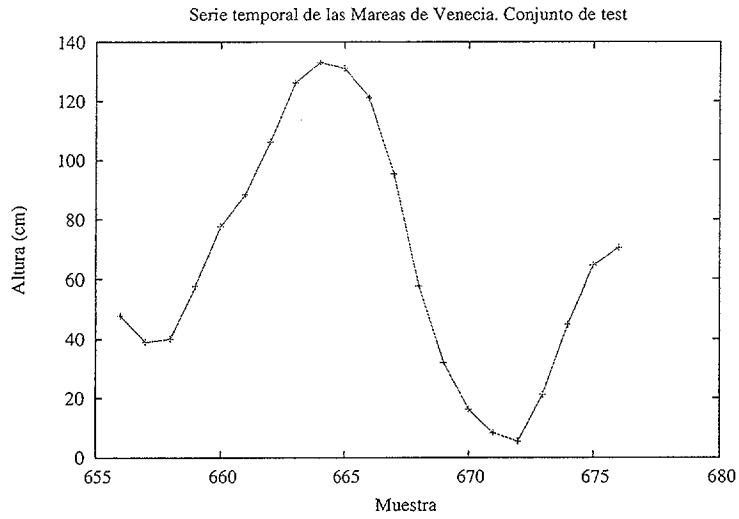


Figura 4.4:
Serie temporal de las Mareas de Venecia. Conjunto de Test

Serie temporal de Mackey-glass

Es una serie temporal caótica generada artificialmente, y es utilizada muy frecuentemente en la literatura relacionada con las RNBR. Entre los trabajos donde se utiliza destacan los siguientes: [Platt, 1991], [Yingwei et al., 1997], [Moody and Darken, 1989] y [Whitehead and Choate, 1995]. Esta serie está generada por la siguiente ecuación diferencial ordinaria, que representa la evolución temporal:

$$\frac{ds(t)}{dt} = -bs(t) + a\frac{s(t-\tau)}{1+s(t-\tau)^{10}} \tag{4.3}$$

donde $a = 0.2, b = 0.1$, y $\tau = 17$ Integrando la ecuación sobre el intervalo de tiempo $[t, t+\Delta t]$ tenemos:

$$x(t+\Delta t) = \frac{2-b\Delta t}{2+b\Delta t}x(t) + \frac{a\Delta t}{2+b\Delta t}\left[\frac{x(t+\Delta t-\tau)}{1+x^{10}((t+\Delta t-\tau)} + \frac{x(t-\tau)}{1+x^{10}(t-\tau)}\right] \tag{4.4}$$

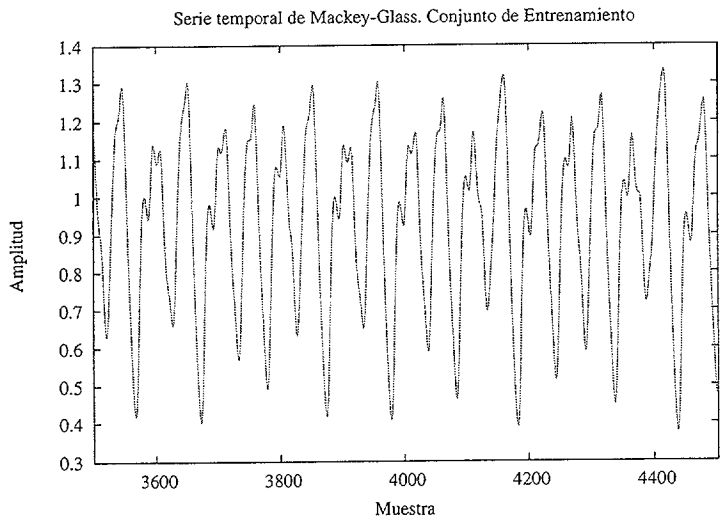


Figura 4.5:
Serie temporal de Mackey-Glass. Conjunto de Entrenamiento

Se establece $\Delta t = 1$, y se genera la serie temporal con la condición inicial: $x(t-\tau) = 0.3$ para los valores de $0 \leq t \leq a\tau$ siendo $\tau = 17$. Se descartan los primeros 3500 datos de la serie para que desaparezcan los transitorios de inicialización. Se predice la muestra n con un horizonte de $\nu = 50$ pasos,

utilizando 4 muestras pasadas: $s_{n-\nu}, s_{n-\nu-6}, s_{n-\nu-12}, s_{n-\nu-18}$. Por lo tanto, la función que la red debe aprender será:

$$x(t) = f(x(t-50), x(t-50-6), x(t-50-12), x(t-50-18)) \quad (4.5)$$

Es decir, el espacio de entrada será de 4 dimensiones.
Para el *conjunto de entrenamiento* se han tomado los 1000 puntos correspondientes al intervalo de tiempo [3500,4499], como puede observarse en la figura 4.5

Para el conjunto de test, se han seleccionado puntos que se encuentran en un intervalo temporal posterior al de entrenamiento, el correspondientes al intervalo de tiempo [4500,5000] (ver figura 4.6).

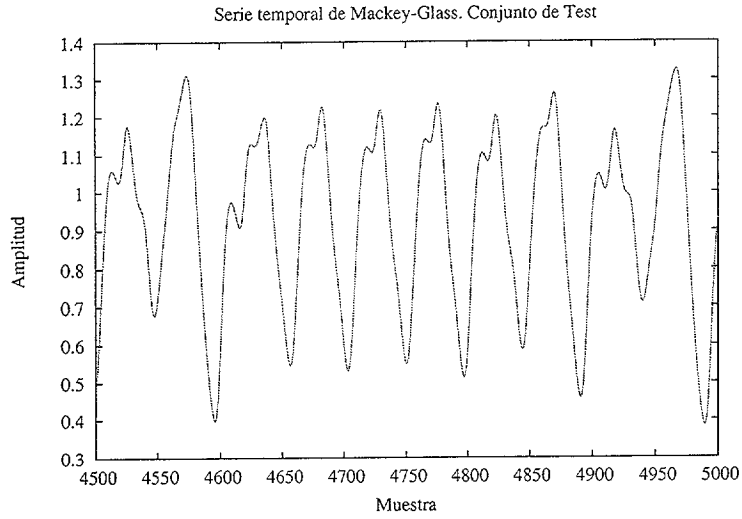


Figura 4.6:
Serie temporal de Mackey-Glass. Conjunto de Test

Al igual que en los dominios anteriores, todos los datos se han normalizado en el intervalo $[0, 1]$.

4.1.3 Dominio de Clasificación

Pima-Indians Diabetes

Es un dominio real sobre enfermos de diabetes de una determinada población, habiéndose obtenido los datos del *UCI Benchmark Repository*:

(http://www.ics.uci.edu/ mlearn/MLSummary.html). Es una base de datos generada por el *National Institute of Diabetes and Digestive and Kidney Diseases*, y es ampliamente utilizada en la literatura de Redes de Neuronas, [Friedman et al., 1997], [Ster and Dobnikar, 1996], [Bennett and Blue, 1997]. Se compone de 768 instancias con 9 atributos numéricos cada una, incluyendo la clase a la que pertenece la instancia. Estos atributos, transcritos literalmente en inglés de la documentación original, son los siguientes:

1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Por tanto, el espacio de entrada será de 8 dimensiones. La distribución de las clases en este conjunto de datos es la siguiente: 500 instancias correspondientes a la clase 0 (65 %), y 268 instancias correspondientes a la clase 1 (35 %). Los valores de los atributos se han normalizado entre 0 y 1.

El *conjunto de entrenamiento* se compone de 576 instancias y el *conjunto de test* se compone de 192 instancias. Esta distribución se ha realizado de forma que se conserve la proporción entre las 2 clases, es decir en ambos conjuntos el 65 % de las instancias pertenecen a la clase 0 y el 35 % restante pertenecen a la clase 1.

4.2 Entrenamiento convencional

En esta etapa de la experimentación se han entrenado RNBR's con diferentes arquitecturas (distinto número de neuronas), por el procedimiento clásico, es decir, se entrena la red con todos los patrones de entrenamiento disponibles y finalizado este entrenamiento, se valida la red con todos los patrones del conjunto de test. El objetivo de esta experimentación es poder comparar los resultados con los obtenidos aplicando el método de aprendizaje selectivo propuesto.
En los dominios de aproximación de funciones o de predicción de series temporales, se ha medido el error medio, $\overline{e}$, obtenido sobre los patrones del correspondiente conjunto de test, para cada arquitectura de red.

$$\overline{e} = \sum_{k=1}^{n} \frac{e_k}{N} \tag{4.6}$$

siendo e_k el error absoluto en el patrón k-ésimo, es decir, el valor absoluto de diferencia entre la salida deseada y la salida que produce la red:

$$e_k = \mid \tilde{y}_k - y_k \mid \tag{4.7}$$

En el dominio de clasificación se ha medido, para cada arquitectura de red, el porcentaje de aciertos, es decir el porcentaje de patrones del conjunto de test que han sido bien clasificados por la red. En el dominio de clasificación estudiado, el número de clases es 2, por lo que a una de ellas se le ha asignado el valor 0 y a la otra el valor 1. Se ha considerado que la red clasifica al patrón como perteneciente a la clase 0, cuando el valor de la salida es inferior a 0.4, y como perteneciente a la clase 1 cuando el valor de salida es superior a 0.6.
A continuación se mostrarán los resultados obtenidos en los diferentes dominios utilizados.

4.2.1 Función Definida por Partes

En este dominio se han entrenado redes desde 10 hasta 130 neuronas, incrementando el número de neuronas de 10 en 10. Se han entrenado las redes durante 500 ciclos de aprendizaje, siendo la tasa de aprendizaje $\eta = 0.05$. En la tabla 4.1 se muestran, los resultados obtenidos por parte de las redes estudiadas. En la figura 4.7 se pueden ver los resultados obtenidos representados gráficamente. Se observa que a partir de 30 neuronas el error obtenido se estabiliza en torno a 0.04.

Neuronas	ErrMedio
10	0.15290
20	0.06766
30	0.05510
40	0.04666
50	0.05287
60	0.04803
70	0.04312
80	0.04418
90	0.04405
100	0.04156
110	0.04496
120	0.04361
130	0.04281

Tabla 4.1:
Errores medios con RNBR tradicional. Función definida por partes

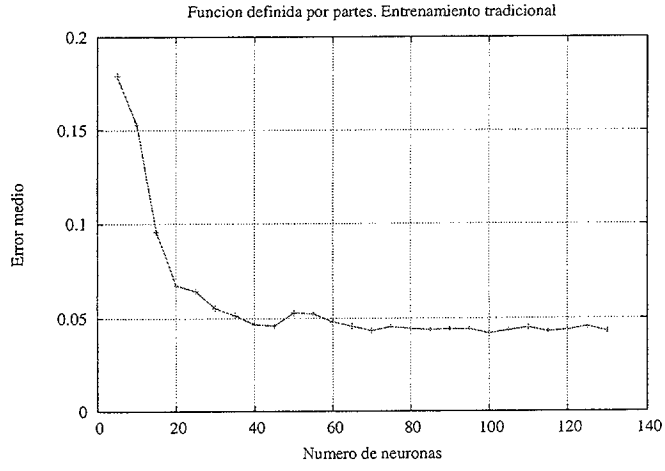


Figura 4.7:
Errores medios con RNBR tradicional. Función definida por partes

4.2.2 Polinomio de Hermite

En este dominio se han entrenado redes de 5, 10, 15, ..., 130 neuronas de forma convencional, de la misma forma que en el dominio anterior. El número de ciclos y la tasa de aprendizaje son los mismos que en el dominio

anterior. En la tabla 4.2 se muestran los resultados obtenidos por parte de las redes estudiadas. En la figura 4.8 se pueden ver los resultados obtenidos representados gráficamente. Se observa que cuando se utilizan redes de más de 20 neuronas el error obtenido se mantiene cercano a 0.025.

Neuronas	ErrMedio
10	0.11569
20	0.02702
30	0.02134
40	0.01904
50	0.02272
60	0.02215
70	0.02066
80	0.02263
90	0.02628
100	0.02145
110	0.02143
120	0.02338
130	0.02508

Tabla 4.2:
Errores medios con RNBR tradicional. Polinomio de Hermite

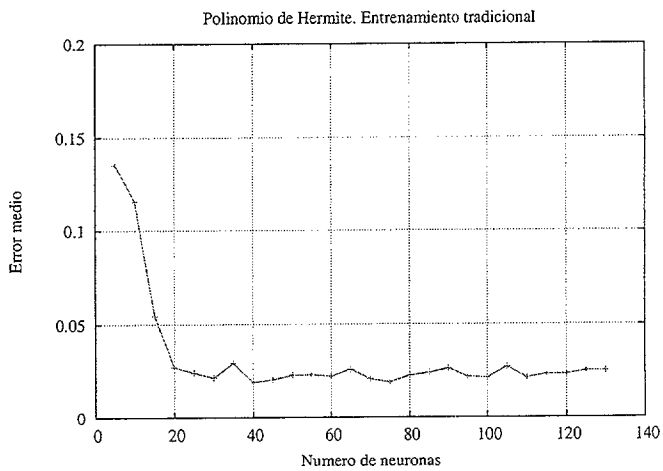


Figura 4.8:
Errores medios con RNBR tradicional. Polinomio de Hermite

4.2.3 Serie temporal de las Mareas de Venecia

El dominio de las mareas de Venecia es de una naturaleza diferente a los anteriores. Como se ha explicado en la subsección 4.1.2, se trata de una serie temporal donde para predecir el valor de la serie en un instante t, se utilizan los valores de la serie en los instantes $t-1$, $t-2$, $t-3$, $t-4$, $t-5$ y $t-6$. Por tanto, se puede considerar el problema de predicción de la serie temporal equivalente al problema de aproximar una función de 6 dimensiones.

En este dominio se han entrenado, redes de 10, 20, ..., 120 neuronas. Se ha utilizado el mismo número de ciclos de entrenamiento y el mismo valor de la tasa de aprendizaje que en los dominios anteriores. En la tabla 4.3 se muestran los resultados obtenidos por parte de las redes estudiadas. En la figura 4.9 se pueden ver los resultados obtenidos representados gráficamente. Se observa que con 10 neuronas el error es muy alto, puesto que son insuficientes para el gran número de patrones de entrenamiento disponibles y para la alta dimensión de los datos de entrada. A medida que el número de neuronas aumenta, el error va bajando, se alcanza el error mínimo, 0.09605, con 50 neuronas y sube muy lentamente cuando el número de neuronas sigue aumentando.

Neuronas	ErrMedio
10	0.23650
20	0.13411
30	0.11173
40	0.11204
50	0.09605
60	0.10290
70	0.10228
80	0.10649
90	0.12137
100	0.12546
110	0.12904
120	0.13802
130	0.14146

Tabla 4.3:
Errores medios con RNBR tradicional. Serie temporal de las Mareas de Venecia

Como se comentó en el apartado 4.1.2 donde se describía el dominio de las Mareas de Venecia, en el conjunto de entrenamiento existen muchos patrones

que representan situaciones normales (periódicas) de marea, mientras que hay muy pocos que representan los fenómenos de "agua alta". En el conjunto de test se han incluido patrones correspondientes a la situación de "agua alta", que es la que interesa predecir.

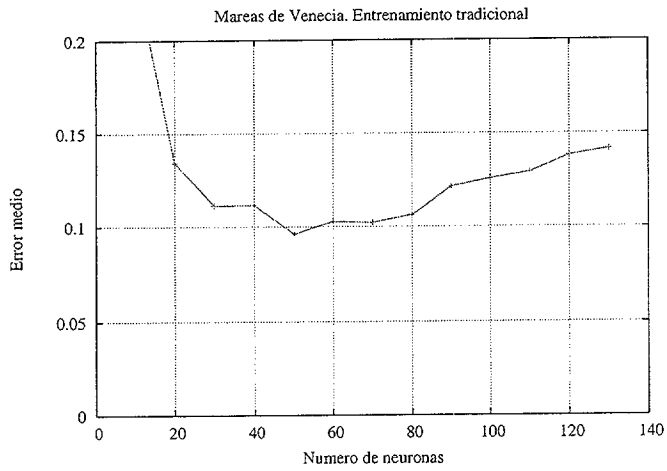


Figura 4.9:
Errores medios con RNBR tradicional. Serie temporal de las Mareas de Venecia

4.2.4 Serie temporal de Mackey-Glass

En este dominio se han entrenado por el método convencional redes de 10, 20, ..., 130 neuronas, habiendo fijado la tasa de aprendizaje y el número de ciclos de entrenamiento a los mismos valores que en los dominios anteriores. En la tabla 4.4 se muestran los resultados obtenidos por todas las redes estudiadas, mostrándose representados gráficamente en la figura 4.10. Se observa que el valor mínimo del error, 0.10273, se alcanza con una arquitectura de 110 neuronas.

Neuronas	ErrMedio
10	0.13296
20	0.13556
30	0.12714
40	0.12768
50	0.11229
60	0.1052
70	0.1274
80	0.11154
90	0.11771
100	0.11628
110	0.10273
120	0.11144
130	0.12768

Tabla 4.4: Error medio con RNBR tradicional. Serie de Mackey-Glass

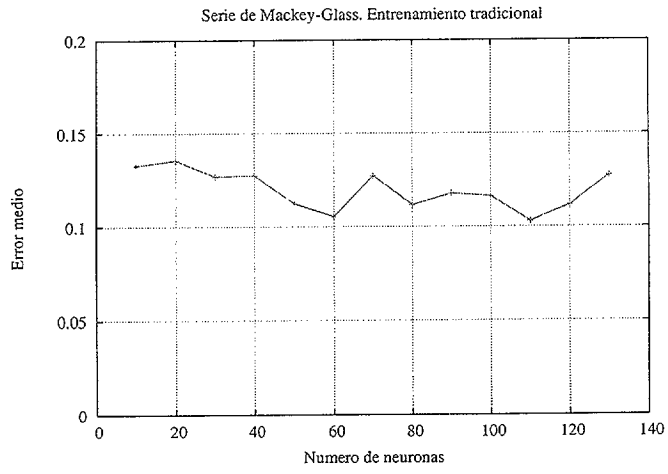


Figura 4.10: Error medio con RNBR tradicional. Serie de Mackey-Glass

4.2.5 Pima Diabetes

En el dominio de clasificación Pima-Indians Diabetes se han entrenado de forma convencional redes de 5, 10, ..., 100 neuronas, manteniendo los parámetros del entrenamiento (tasa de aprendizaje y número de ciclos) en los mismos valores que en los casos anteriores. En la tabla 4.5 se mues-

tran los resultados obtenidos por las redes estudiadas, mostrando la tasa de aciertos obtenida para todos los patrones del conjunto de test, es decir, la proporción de patrones clasificados correctamente sobre el total.

Neuronas	Tasa Aciertos
10	0.72396
20	0.73958
30	0.73438
40	0.72917
50	0.73274
60	0.72917
70	0.67708
80	0.62501
90	0.59375
100	0.51563

Tabla 4.5: Tasa de Aciertos con RNBR tradicional. Pima-Indians Diabetes

En la figura 4.11 se pueden ver los resultados obtenidos representados gráficamente. Se observa que los valores de la tasa de aciertos se mantiene relativamente estable en torno al 73%, bajando ese porcentaje cuando las redes superan las 70 neuronas.

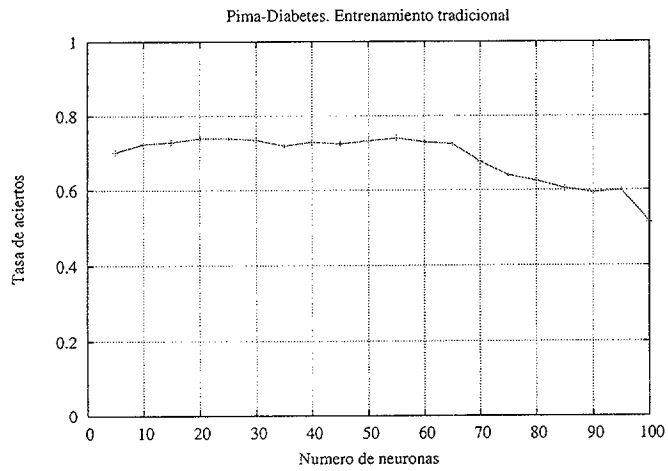


Figura 4.11: Tasa de aciertos con RNBR tradicional. Pima-Indians Diabetes

4.3 Ponderación Gaussiana

El método propuesto en esta tesis, para entrenar RNBR, consiste en seleccionar del conjunto completo de entrenamiento, un subconjunto apropiado de patrones con el objetivo de mejorar la respuesta de la red a cada nuevo patrón de test. En este subconjunto aparecerían aquellos patrones más cercanos, en términos de distancia euclídea, al patrón de test. Además, los patrones seleccionados pueden repetirse en el subconjunto, de forma que cuanto más cerca se encuentren del patrón de test, más veces se repetirán. De esta forma, la red se entrenará con la información más útil, descartando aquellos patrones que no sólo no proporcionan conocimiento a la red, sino que además pueden distorsionar el proceso de aprendizaje.

El planteamiento central del método propuesto consiste en asignar un peso a cada patrón de entrenamiento, en función de la distancia de este punto al patrón de test. Esta función de ponderación, o función kernel, K, debe tener las siguientes características:

- Su máximo valor debe darse cuando la distancia es 0.
- El valor de la función debe disminuir sin discontinuidades a medida que la distancia aumenta.

En esta primera propuesta del método, denominada *Ponderación Gaussiana* y a la que en lo sucesivo se hará referencia con el nombre de *propuesta 1*, se ha utilizado como función Kernel la función gaussiana o función normal. Esta función tiene la siguiente expresión analítica:

$$K(\mathbf{x_k}) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{d(q,\mathbf{x_k})^2}{2\sigma^2}} \tag{4.8}$$

donde $d(q,\mathbf{x_k})^2$ es la distancia euclídea entre el patrón de test q y el patrón de entrenamiento $\mathbf{x_k}$, y σ es un parámetro llamado desviación. Para simplificar la notación, esta distancia se representará por d_k.

Del valor real de la función obtenemos un número entero n_k que indicará el número de veces que el patrón $\mathbf{x_k}$ será incluido en el conjunto de entrenamiento. Dicho valor viene dado por la parte entera de $K(\mathbf{x_k})$, es decir: $n_k = int(K(\mathbf{x_k}))$

En la figura 4.12 pueden verse dos representaciones de la función kernel para un conjunto de patrones unidimensionales representados en el eje de abcisas, tomando como patrón de test el punto $q = 2.25$. La gráfica A corresponde a una desviación $\sigma = 0.2$ y la gráfica B a $\sigma = 0.6$. Como puede

observarse al comparar ambas gráficas, cuanto menor es la desviación (curva A) más estrecha y más alta es la "campana", de forma que el valor de la función para puntos muy próximos a q es muy grande y decrece rápidamente cuando los puntos se alejan de q. Por tanto, con desviaciones pequeñas (curva A) sólo los puntos muy próximos al patrón de test se incluirán en el subconjunto de patrones seleccionados, repitiéndose un gran número de veces. Sin embargo, con desviaciones grandes (curva B), se seleccionarán más patrones aunque el número de veces que se repetirán será menor.

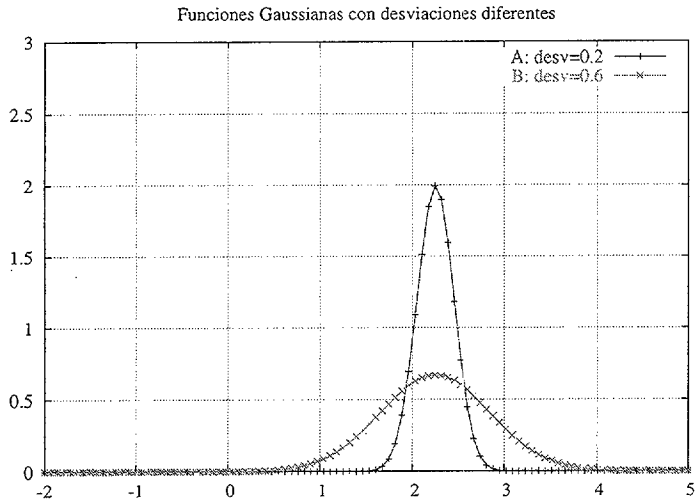


Figura 4.12: Función Gaussiana como función kernel

4.3.1 Descripción del método

A continuación se describe el método de forma detallada.

Sea $\mathbf{q}$ un patrón de test descrito por un vector n-dimensional, $\mathbf{q} = (q_1, ..., q_n)$, donde q_i representa los atributos de la instancia $\mathbf{q}$. Sea X el conjunto disponible de patrones de entrenamiento:

$$X = \{(x_i, y_i), i = 1, ..., N; x_i = (x_{i1}, ...x_{in}); y_i = (y_{i1}, ...y_{im})\} \qquad (4.9)$$

donde x_i son las entradas de los patrones e y_i sus respectivas salidas. Los pasos a seguir para seleccionar el conjunto de entrenamiento asociado al patrón $\mathbf{q}$, llamado X_q , son los siguientes:

1. Se asocia a cada patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ un valor real d_k, que se define en términos de la distancia euclídea desde el patrón de test $\mathbf{q}$ a cada patrón de entrenamiento. De forma más precisa se define como:

$$d_k = d(\mathbf{x_k}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(x_{ki} - q_i)^2} \qquad (4.10)$$
$$k = 1, 2...N$$

   Esta distancia proporciona una medida para establecer los patrones de entrenamiento más cercanos al patrón de test. Dicho de otra forma, los patrones de entrenamiento se ponderan para poder replicar los patrones relevantes y descartar los irrelevantes, tal como se describe en los pasos siguientes.

2. Se asocia a cada patrón $(\mathbf{x_k}, \mathbf{y_k})$ el valor real resultante de aplicar la función Kernel K a la distancia d_k:

$$K(\mathbf{x_k}) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{d_k^2}{2\sigma^2}}, \text{ para } k = 1, 2, \ldots, N \qquad (4.11)$$

3. Los valores $K(\mathbf{x_k})$, previamente calculados, serán utilizados para indicar cuántas veces el patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ será repetido en el nuevo subconjunto de entrenamiento. Por tanto, estos valores deberán ser transformados en números enteros. La forma más intuitiva de hacer esta transformación consiste en tomar la parte entera del número real $K(\mathbf{x_k})$:

$$n_k = int(K(\mathbf{x_k})) \qquad (4.12)$$

   En este punto, cada patrón de entrenamiento en X tiene un número natural n_k asociado que indica cuántas veces el patrón $(\mathbf{x_k}, \mathbf{y_k})$ debe ser utilizado para entrenar la RNBR cuando llegue la nueva instancia q.

4. De este modo se construye un nuevo subconjunto de patrones de entrenamiento X_q asociado al patrón de test q. Dado un patrón $(\mathbf{x_k}, \mathbf{y_k})$ perteneciente al conjunto original X, se incluirá en el nuevo subconjunto X_q si el valor n_k es mayor que 0. Además, ese patrón será repetido aleatoriamente n_k veces en el subconjunto X_q.

5. Una vez seleccionados los patrones de entrenamiento, la RNBR se entrena con el subconjunto X_q. Como se mencionó en el capítulo

2.1, el entrenamiento de una RNBR supone la determinación de los centros de las funciones de activación Gaussianas, las desviaciones y los pesos. Los centros se calculan en modo no supervisado utilizando el algoritmo K-medias para clasificar el espacio de entrada, formado por los patrones seleccionados sin repetir, es decir, por los patrones del conjunto W_q, siendo $W_q \subseteq X_q$, el conjunto resultante de extraer los patrones repetidos del conjunto X_q. A continuación, se calculan los coeficientes de desviación de las neuronas como la raíz cuadrada del producto de las distancias de su centro a los centros de las neuronas más próximas. Por último, se calculan los pesos de las conexiones de la red de forma que se minimice el error cuadrático medio sobre los patrones del conjunto X_q.

4.3.2 Resultados Experimentales

Se ha aplicado el método descrito anteriormente a los siguientes dominios:

- Función Definida por Partes
- Polinomio de Hermite
- Serie temporal de Mackey Glass
- Serie temporal de las Mareas de Venecia

En todos ellos se han utilizado arquitecturas de RNBR de 5, 9, 13 y 17 neuronas, y los siguientes valores del parámetro σ o desviación de la función Kernel: 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35 y 0.4. Además, se han utilizado en todos los casos los siguientes parámetros en el entrenamiento de las redes: tasa de aprendizaje $\eta = 0.05$ y 500 ciclos de aprendizaje.

Para cada patrón de test, se selecciona un conjunto de entrenamiento X_q, utilizando un determinado valor del parámetro σ con el que se entrena la red de una arquitectura dada y se mide el error de test como el valor absoluto de la diferencia entre la salida y la salida deseada para ese patrón de test. Una vez que se ha aplicado a todos los patrones de test, se calcula el error medio como la media aritmética de los errores de test. Este proceso se repite para cada arquitectura y cada valor de σ. Los resultados obtenidos para cada uno de los dominios son los siguientes:

Función Definida por Partes

En la tabla 4.6 se muestran los valores de los errores medios obtenidos.

	Neuronas Ocultas			
Desviación	5	9	13	17
0.01	0.0448	0.0996	0.1787	0.2589
0.05	0.0240	0.0608	0.1251	0.2078
0.1	0.0259	0.0361	0.1405	0.2331
0.15	0.0315	0.0288	0.0798	0.1327
0.2	0.0360	0.0259	0.0514	0.1730
0.25	0.0398	0.0233	0.0628	0.1495
0.3	0.0328	0.0245	0.0677	0.1552
0.35	0.0268	0.0363	0.1396	0.2018
0.4	0.4723	0.4804	0.4839	0.4811

Tabla 4.6:
Errores medios con aprendizaje selectivo (Propuesta 1). Función Definida por Partes

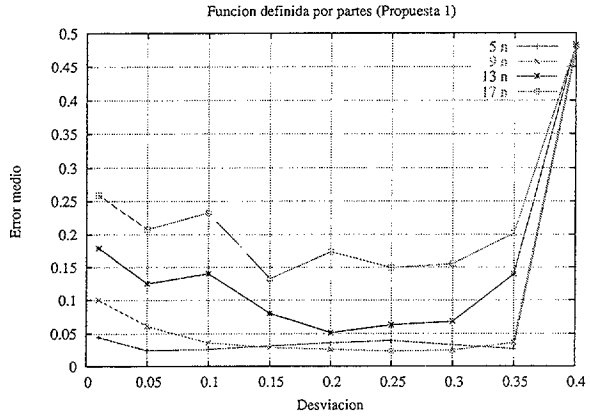


Figura 4.13:
Errores medios con aprendizaje selectivo (Propuesta 1). Función Definida por Partes

En la figura 4.13 se muestra la representación gráfica de los resultados de la tabla 4.6, y se observa lo siguiente:

Cuando la desviación es muy pequeña, $\sigma \leq 0.05$, el error es relativamente grande, aunque se comportan mejor las redes de menor número de neuronas. Con $\sigma = 0.01$ la red de 5 neuronas alcanza un error de 0.0448 mientras que la de 17 neuronas alcanza un error de 0.2589, casi 6 veces mayor. Esto se explica porque con desviaciones muy pequeñas, la campana es muy estrecha y alta, es decir, el número de patrones seleccionados es muy pequeño, aunque se

repiten muchas veces. Si el número de neuronas es grande, la red se comporta mal con muy pocos patrones, aunque éstos estén repetidos un gran número de veces, mientras que una red de pocas neuronas se ajusta mejor a un espacio de entrada tan reducido. El error va disminuyendo a medida que σ aumenta, porque el número de patrones seleccionados también aumenta, manteniéndose relativamente estable hasta desviaciones cercanas a 0.35. A partir de este valor de σ el error aumenta mucho en todas las arquitecturas. Esto es debido a que al ensancharse mucho la curva y bajar su valor máximo, el valor real de la función K para la mayoría de los patrones es menor que uno y por tanto, el valor entero que indicará el número de veces que se repetirá el patrón se hace cero.

Polinomio de Hermite

Se ha aplicado el método descrito al dominio del Polinomio de Hermite, para las mismas arquitecturas y desviaciones que en el dominio anterior, habiéndose obtenido como resultado los errores medios mostrados en la tabla 4.7. Estos resultados están representados gráficamente en la figura 4.14 observándose un comportamiento similar al de la Función definida por partes. Las redes con menor número de neuronas consiguen errores más pequeños que las de mayor número de neuronas. Además, en casi todos los casos los errores disminuyen ligeramente a medida que aumenta la desviación, y aumentan bruscamente cuando ésta es superior a 0.35. Al igual que ocurría en el anterior dominio, cuando σ es muy pequeña la campana de Gauss es muy alta y estrecha, de forma que se seleccionan muy pocos patrones, aunque éstos se repitan un gran número de veces. Al entrenarse las redes con muy pocos patrones, consiguen un mejor rendimiento las que tienen menor número de neuronas por adaptarse mejor a un espacio de entrada tan reducido. Si la desviación va aumentando, la campana se ensancha y baja, seleccionándose un mayor número de patrones de entrenamiento; por este motivo se observa que el error va mejorando, aunque si sigue aumentando la desviación, la campana se ensancha más pero también disminuye el valor de la función llegando a alcanzar en zonas más amplias del espacio de entrada valores inferiores a uno, por lo que los patrones correspondientes no serán seleccionados, ya que el valor $n_k = int(K(\mathbf{x_k}))$ será cero; dicho de otra forma, llega un momento en que al aumentar la desviación vuelven a seleccionarse menos patrones de entrenamiento, además repetidos un número muy pequeño de veces; si se segue aumentando σ, el valor máximo de la campana será inferior a uno, con lo cual ningún patrón podrá ser seleccionado y el error de la red será muy alto, como se puede observar en la gráfica.

Se destaca que el mejor valor de error obtenido es 0.0240, obtenido por una red de 5 neuronas con una desviación $\sigma = 0.05$.

	Neuronas Ocultas			
Desviación	5	9	13	17
0.01	0.0982	0.2182	0.3432	0.3821
0.05	0.0283	0.1633	0.3238	0.3849
0.1	0.0121	0.1180	0.2352	0.3891
0.15	0.0134	0.0944	0.2113	0.3519
0.2	0.0133	0.0738	0.2133	0.2672
0.25	0.0123	0.0566	0.1832	0.3044
0.3	0.0134	0.0887	0.1923	0.2920
0.35	0.0135	0.0973	0.2398	0.3412
0.4	0.4513	0.4492	0.4457	0.4321

Tabla 4.7:
Errores medios con aprendizaje selectivo (Propuesta 1). Polinomio de Hermite

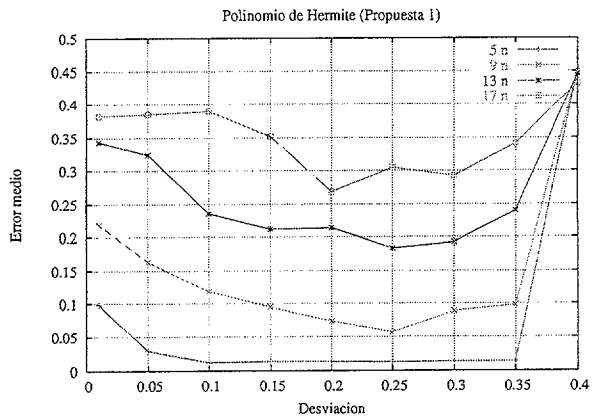


Figura 4.14:
Errores medios con aprendizaje selectivo (Propuesta 1). Polinomio de Hermite

Serie temporal de las Mareas de Venecia

Se ha aplicado el método al dominio correspondiente a la serie temporal de las Mareas de Venecia, que tiene características diferentes a los dominios anteriores. Al haberse decidido tomar 6 valores de la serie como entrada pa-

ra predecir el valor en un instante posterior, la función que la red intentará aprender será un función de 6 dimensiones; por tanto, los patrones de entrenamiento y de test serán puntos en un espacio de 6 dimensiones mientras que los patrones correspondientes a los dominios anteriores eran unidimensionales. Esto hará que las distancias entre patrones sean mayores. Por otra parte, como se dijo en el apartado donde se describía el dominio de las Mareas de Venecia, hay muchos patrones que representan situaciones normales de marea mientras que hay pocos patrones que representan las situaciones excepcionales de 'agua alta'. Es decir, en el espacio de entrada habrá zonas de alta densidad de patrones (representando situaciones normales) y otras de baja densidad (representando situaciones de agua 'alta'), en las cuales para un patrón de test, los patrones de entrenamiento más próximos estarán relativamente alejados. Por tanto, los datos no están uniformemente distribuidos en el espacio de entrada, al contrario de lo que ocurría con los patrones de los dominios anteriores que sí estaban uniformemente distribuidos. Estas diferencias tendrán importancia en el comportamiento de las redes. En la tabla 4.8 se muestran los datos obtenidos para las mismas desviaciones y arquitecturas que en los dominios anteriores.

	Neuronas Ocultas			
Desviación	5	9	13	17
0.01	0.6344	0.6344	0.6344	0.6344
0.05	0.1490	0.1315	0.1324	0.1350
0.1	0.1059	0.1120	0.1255	0.1259
0.15	0.1239	0.1180	0.1072	0.0905
0.2	0.1300	0.1155	0.1164	0.1068
0.25	0.1342	0.1191	0.1138	0.1118
0.3	0.1298	0.1170	0.1115	0.1337
0.35	0.1023	0.1134	0.1332	0.1357
0.4	0.6344	0.6344	0.6344	0.6344

Tabla 4.8:
Errores medios con aprendizaje selectivo (Propuesta 1). Serie temporal de las Mareas de Venecia

En la figura 4.15 se representan gráficamente los datos anteriores. Puede verse que en la zona del eje de abcisas correspondiente a desviaciones comprendidas entre 0.05 y 0.35 los errores medios se mantienen relativamente cercanos al valor 0.1. Además, no hay diferencias significativas entre las diferentes arquitecturas, pues todas ellas obtienen valores similares. Cuando la desviación es menor de 0.05 o mayor de 0.35 los errores son muy altos

en todos los casos. Esto se explica por el hecho de que en ningún caso se seleccionan patrones para entrenar las redes: cuando la desviación es muy pequeña, la campana es muy alta y estrecha, de forma que sólo los patrones de entrenamiento que estén muy cerca de la muestra de test podrán ser seleccionados. Pero en este espacio de 6 dimensiones las distancias son más grandes, de forma que ningún patrón es seleccionado. Cuando las desviaciones son grandes, la campana es ancha y baja, correspondiendo el valor de la función $K(\mathbf{x_k})$ para todos los patrones de entrenamiento a valores inferiores a uno. Por tanto, los valores n_k de todos los patrones de entrenamiento serán nulos y no serán seleccionados.

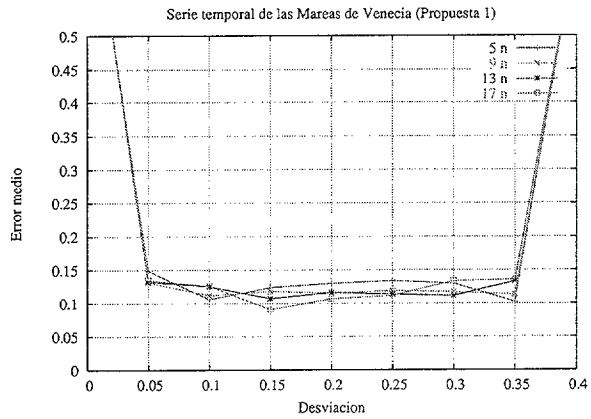


Figura 4.15:
Errores medios con aprendizaje selectivo (Propuesta 1). Serie temporal de las Mareas de Venecia

Serie temporal de Mackey-Glass

Se ha aplicado el método propuesto al dominio de Mackey-Glass, que tiene características similares al dominio de las Mareas de Venecia, por la alta dimensionalidad y la distribución de los patrones en el espacio de entrada. La alta dimensionalidad se debe a la decisión de tomar 4 valores de la serie para predecir el valor de ésta en un instante posterior. Por este motivo, la función a aprender por la red será una función de 4 dimensiones, por lo que las distancias entre patrones son relativamente grandes. Además, los patrones tampoco están uniformemente distribuidos en el espacio de entrada: existen zonas de este espacio donde la densidad de patrones es alta mientras que en otras zonas el espacio está casi vacío. En la tabla 4.9 pueden verse

los valores de los errores medios obtenidos para las mismas arquitecturas y desviaciones utilizadas en los dominios anteriores.

	Neuronas Ocultas			
Desviación	5	9	13	17
0.01	0.3047	0.3138	0.3217	0.3376
0.05	0.0463	0.0467	0.0622	0.0808
0.1	0.0434	0.0349	0.0386	0.0466
0.15	0.0535	0.0381	0.0359	0.0393
0.2	0.0652	0.0398	0.0374	0.0411
0.25	0.0664	0.0425	0.0355	0.0406
0.3	0.0647	0.0417	0.0377	0.0413
0.35	0.0605	0.0427	0.0417	0.0511
0.4	0.5752	0.5752	0.5752	0.5752

Tabla 4.9:
Errores medios con aprendizaje selectivo (Propuesta 1). Serie temporal de Mackey-Glass

Estos resultados se han representado gráficamente en la figura 4.16. En ella se observa un comportamiento muy similar al de las Mareas de Venecia: existe una zona con $0.05 \leq \sigma \leq 0.35$ donde el error alcanza valores mínimos. Además, los comportamientos de las redes de diferentes arquitecturas son muy similares y apenas hay diferencias entre las redes de 5 y las de 17 neuronas. Cuando las desviaciones son muy pequeñas, el error crece mucho, aunque no tanto como en el dominio de las Mareas de Venecia. Esto es debido a que ciertos patrones de test estarán ubicados en zonas del espacio de entrada con una gran densidad de patrones, y por tanto, algunos patrones de entrenamiento serán seleccionados, aunque debido a la complejidad de la serie temporal de Mackey-Glass, la red necesita más ejemplos de entrenamiento para realizar un buen ajuste. Por eso, al ser la desviación muy pequeña, se seleccionan muy pocos patrones, siendo insuficientes para entrenar adecuadamente la red. Por tanto, los errores obtenidos en este rango serán grandes, aunque no tanto como en el dominio anterior.

Al igual que ocurría en los dominios anteriores, el error crece mucho cuando la desviación es mayor que 0.35. El motivo es el mismo: la curva es más 'achatada' y los valores de la función K son menores que uno, incluso para los patrones más cercanos al patrón de test, no siendo por tanto seleccionados.

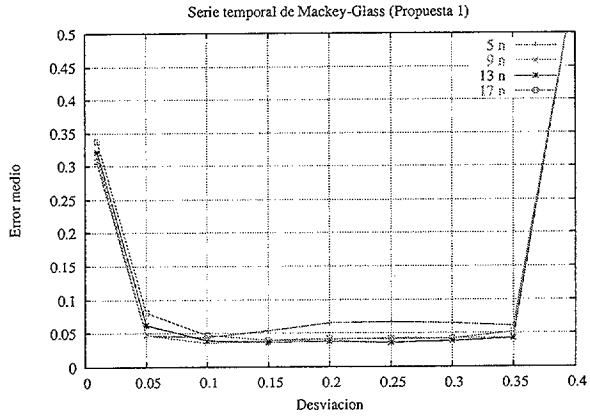


Figura 4.16:
Errores medios con aprendizaje selectivo (Propuesta 1). Serie temporal de Mackey-Glass

Comparación con el método tradicional

En la tabla 4.10 se comparan los mejores resultados obtenidos por el método propuesto utilizando ponderación gaussiana, es decir cuando para cada patrón de test se seleccionan los patrones de entrenamiento más cercanos a él utilizando la función gaussiana, con los mejores resultados alcanzados cuando se entrena una RNBR de forma convencional. Se observa en todos los casos un mejor comportamiento del método propuesto respecto al tradicional, aunque esta mejora varía según los dominios: es más importante en la serie de Mackey-Glass y en la Función Definida por Partes, y muy pequeña en el Polinomio de Hermite y en la serie de las Mareas de Venecia, aunque en todos los casos el número de neuronas necesario es sensiblemente menor.

4.3.3 Conclusiones

Después de analizar los resultados experimentales obtenidos en la Función Definida por Partes, el Polinomio de Hermite y las series temporales de las Mareas de Venecia y de Mackey-Glass, se pueden sacar las siguientes conclusiones:

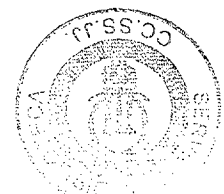
- Con el entrenamiento selectivo, se mejoran ligeramente los resultados obtenidos por las RNBR entrenadas de forma convencional. Esto es debido a que la red se entrena solamente con los patrones más similares al nuevo patrón de test. Aunque el coste computacional es mayor

Error Medio	Entrenamiento selectivo	Entrenamiento tradicional
Función Definida por Partes	0.0233 $\sigma = 0.25$, 9 neuronas	0.04156 100 neuronas
Polinomio de Hermite	0.0121 $\sigma = 0.1$, 5 neuronas	0.01904 40 neuronas
Serie de las Mareas de Venecia	0.0905 $\sigma = 0.15$, 17 neuronas	0.09605 50 neuronas
Serie de Mackey-Glass	0.0348 $\sigma = 0.1$, 9 neuronas	0.10273 110 neuronas

Tabla 4.10:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo y con entrenamiento tradicional

porque hay que entrenar la red para cada patrón de test, el número de neuronas necesario es sensiblemente menor.

- El error depende mucho de la desviación, obteniéndose resultados aceptables dentro de un margen relativamente estrecho de σ. El error es grande con desviaciones muy pequeñas debido a que, al ser la gaussiana muy estrecha, se seleccionan muy pocos patrones de entrenamiento, y en algunos casos ninguno, y la red no es capaz de generalizar adecuadamente. Sin embargo, al crecer la desviación, al principio se seleccionan más patrones, puesto que la curva se ensancha, pero si sigue creciendo σ, llega un momento en que el valor de la función $K(\mathbf{x_k})$ para gran parte de los patrones llega a ser menor que 1, por lo que bruscamente dejan de seleccionarse la mayor parte de los patrones de entrenamiento. Por este motivo, el error aumenta de forma muy acusada.



4.4 Ponderación Inversa

En la propuesta anterior se utilizaba una función gaussiana como función kernel para seleccionar patrones. Los resultados obtenidos son mejores que los de las redes entrenadas por el método tradicional, pero como puede verse en los resultados mostrados en la sección anterior, el error medio es muy dependiente del parámetro σ de la función gaussiana. Este hecho es un inconveniente del método, pues es necesario determinar el valor adecuado de la desviación, ya que de lo contrario el error puede ser mucho mayor que el que se obtendría con el método convencional.
En esta propuesta, denominada *Ponderación Inversa*, se modifica el método descrito en 4.3 para tratar de evitar esta gran dependencia de la desviación. Se modificará la función kernel, y en lugar de utilizar una función gaussiana, se utilizará una función inversa:

$$K(\mathbf{x_k}) = \frac{1}{d_k} \tag{4.13}$$

siendo d_k, la distancia euclídea entre q (el patrón de test) y $\mathbf{x_k}$ (el k-ésimo patrón de entrenamiento). La función inversa cumple la principal característica que debe tener una función kernel, decrece de forma continua a medida que la distancia aumenta, y además no depende de ningún parámetro, al contrario que la función gaussiana. En la figura 4.17 puede verse representada dicha función para el caso de patrones de una dimensión, donde $q = 2.25$. Se observa que al no existir el parámetro desviación, los patrones más cercanos siempre se van a incluir un gran número de veces en el conjunto de entrenamiento.

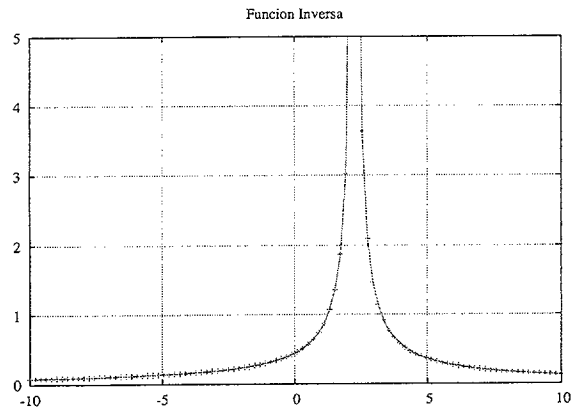


Figura 4.17: Función Inversa como función kernel

Este método de ponderación inversa ha dado lugar a cuatro propuestas. En la primera, que será referenciada como *propuesta 2.1*, es necesario introducir un parámetro al método (no a la función) llamado *valor de redondeo* que se utilizará para transformar los valores reales de la función inversa en los valores enteros n_k que indicarán cuántas veces serán repetidos los patrones seleccionados en el conjunto de entrenamiento X_q. Al analizar esta propuesta, se observan ciertos inconvenientes relacionados con el valor de redondeo, y con el funcionamiento del algoritmo K-medias.

En la segunda propuesta, referenciada como *propuesta 2.2*, se sustituye el valor de redondeo por un valor umbral o *corte* y se modifica la inicialización del algoritmo K-medias, consiguiendo solucionar los inconvenientes anteriores aunque se observa que el comportamiento de las redes depende en gran medida de la inicialización aleatoria del algoritmo K-medias, con lo que se vuelve a modificar el método, pasando a la tercera propuesta, llamada *propuesta 2.3* que incorpora una inicialización determinista de K-medias.

Al aplicar el método a dominios de alta dimensionalidad, donde existen zonas del espacio de entrada con una densidad de patrones de entrenamiento muy baja, se observa que en ocasiones, con cortes muy pequeños, no se selecciona ningún patrón. Para solucionar este problema se incorpora una mejora al método, dando lugar a la cuarta propuesta, llamada *propuesta 2.4*, para que siempre se seleccionen patrones de entrenamiento, aunque en la vecindad del patrón de test no existan a la distancia indicada por el parámetro corte.

4.4.1 Utilización del *valor de redondeo* para determinar la frecuencia

En esta versión del método, a la que se denominará *propuesta 2.1*, se obtiene el valor de la función, $K(\mathbf{x_k})$, asociada a cada patrón $(\mathbf{x_k}, \mathbf{y_k})$ como la inversa de la distancia euclídea absoluta entre el patrón de test $\mathbf{q}$ y cada patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$. Los valores de la función se normalizan, de forma que su suma sea igual al número total de patrones de entrenamiento, obteniéndose el valor llamado frecuencia normalizada f_{nk}. Este valor se utilizará para indicar cuántas veces el patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ se repetirá en el nuevo subconjunto de entrenamiento. Por tanto, estos valores $f'_{nk}s$ deberán transformarse en números naturales. La forma más intuitiva de hacer esta transformación consiste en tomar la parte entera del valor real f_{nk}. Sin embargo, se ha considerado conveniente introducir un parámetro - llamado *valor de redondeo*, c - cuyo valor estará comprendido entre cero y uno, y es utilizado por el método para transformar los valores reales f_{nk} en los valores naturales n_k. Los motivos por los que se ha decidido utilizar este parámetro en lugar de utilizar solamente la parte entera, se explican en el siguiente apartado.

Descripción del método

De la misma forma que en el apartado 4.3.1, donde se describe el método correspondiente a la ponderación gaussiana, se considera $\mathbf{q}$ un patrón de test representado por un vector n-dimensional, $\mathbf{q} = (q_1, ..., q_n)$, donde q_i representa los atributos de la instancia $\mathbf{q}$. Sea X el conjunto disponible de patrones de entrenamiento:

$$X = \{(\mathbf{x_i}, \mathbf{y_i}), i = 1, ..., N; \mathbf{x_i} = (x_{i1}, ...x_{in}); \mathbf{y_i} = (y_{i1}, ...y_{im})\} \qquad (4.14)$$

donde $\mathbf{x_i}$ son las entradas de los patrones e $\mathbf{y_i}$ sus respectivas salidas. Los pasos a seguir para seleccionar el conjunto de entrenamiento asociado al patrón $\mathbf{q}$, llamado X_q , son los siguientes:

1. Se asocia a cada patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ un valor real d_k, que se define en términos de la distancia euclídea desde el patrón de test $\mathbf{q}$ a cada patrón de entrenamiento. Esta distancia es la misma que la descrita en la ecuación 4.10, y proporciona una medida para establecer los patrones de entrenamiento más cercanos al patrón de test, ponderándolos para poder replicar los patrones relevantes y descartar los irrelevantes, tal como se describe en los pasos siguientes.

2. Se asocia a cada patrón $(\mathbf{x_k}, \mathbf{y_k})$ el valor resultante de aplicar la función inversa a la distancia d_k:

$$K(x_k) = \frac{1}{d_k}, k = 1, 2, \ldots, N \tag{4.15}$$

Estos valores $K(x_k)$ son mayores cuanto más cercanos están los patrones $\mathbf{x_k}$ de $\mathbf{q}$, y menores cuanto más alejados están.

3. Se introduce un factor de normalización V para transformar los valores $K(x_k)$, en otros valores $K_n(\mathbf{x_k}) = VK(\mathbf{x_k})$ de forma que su suma sea igual al número total de patrones de entrenamiento:

$$\sum_{k=1}^{N} K_n(x_k) = N \tag{4.16}$$

Para simplificar la notación, en adelante se llamará frecuencia relativa f_{nk} a este valor normalizado $K_n(x_k)$ asociado a cada patrón $\mathbf{x_k}$. El factor de normalización V se calcula del siguiente modo:

$$V = \frac{N}{\sum_{k=1}^{N} K(\mathbf{x_k})} \tag{4.17}$$

De este modo,

$$\sum_{k=1}^{N} f_{nk} = N \tag{4.18}$$

4. Los valores f_{nk} previamente calculados se utilizarán para indicar cuántas veces será repetido el patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ en el nuevo subconjunto de entrenamiento X_q, y por tanto deben ser transformados en números naturales. Este paso es muy similar a la transformación de $K(\mathbf{x_k})$ en n_k en el método de ponderación gaussiana (apartado 4.3.1), y la forma más intuitiva de hacerlo sería tomando la parte entera del valor real f_{nk}. Sin embargo, siguiendo este criterio, aquellos patrones cuyos valores f_{nk} estén incluidos en el intervalo $[0, 1)$ serán descartados y esto no parece muy razonable en determinados casos, pues puede ocurrir que haya zonas del espacio de entrada con poca densidad de patrones de entrenamiento, donde los patrones más cercanos están a distancias relativamente grandes, de modo que su valor asociado $f_{nk} \in [0, 1)$, y por tanto serían descartados. Se ha considerado conveniente introducir un *valor de redondeo* c, como parámetro

para determinar qué valor de frecuencia debe ser considerado cero. Este parámetro, cuyo valor estará comprendido entre 0 y 1, es utilizado por el método para transformar los valores reales f_{nk} en los valores naturales n_k, de la siguiente forma:

$$\begin{array}{lll} \text{Si } [f_{nk} - int(f_{nk})] \geq c, & \text{entonces} & n_k = int(f_{nk}) + 1 \\ \text{Si } [f_{nk} - int(f_{nk})] < c, & \text{entonces} & n_k = int(f_{nk}) \end{array} \tag{4.19}$$

En la propuesta anterior se controlaba el número de patrones seleccionados con el parámetro σ; sin embargo, en la función inversa no existe ningún parámetro, por lo que es conveniente introducir el valor de redondeo c para tener cierto control sobre la selección de patrones. En este punto, cada patrón de entrenamiento en X tiene asociado un número natural n_k que indicará cuántas veces el patrón $(\mathbf{x_k}, \mathbf{y_k})$ será utilizado para entrenar la RNBR cuando el patrón de test sea presentado.

5. Se construye un nuevo conjunto de entrenamiento X_q asociado al patrón de test $\mathbf{q}$. Dado un patrón $(\mathbf{x_k}, \mathbf{y_k})$ perteneciente al conjunto de entrenamiento original X, será incluido en el nuevo conjunto X_q si el valor n_k es mayor que 0. Además, el patrón $(\mathbf{x_k}, \mathbf{y_k})$ será colocado n_k veces de forma aleatoria en el conjunto X_q.

6. Una vez seleccionados los patrones de entrenamiento, se entrena la red con el nuevo conjunto X_q. al igual que se hacía en la fase 1, se genera un conjunto inicial de entrenamiento W_q con los patrones seleccionados repetidos una sola vez, es decir aquellos patrones $\mathbf{x_k}$ tales que $n_k > 0$. Con este conjunto se determinarán los centros de las neuronas utilizando el algoritmo K-medias, que se inicializa generando k puntos aleatorios en el espacio de entrada. Una vez determinados los centros, se calcularán las desviaciones de cada neurona como la raíz cuadrada del producto de las distancias a los dos centros más próximos, y se utilizará el conjunto de entrenamiento X_q, con los patrones repetidos, para ajustar los pesos de la red.

Se ha aplicado este método a los dominios estudiados en la propuesta anterior, Función definida por partes y Serie temporal de Mackey-Glass, así como al Polinomio de Hermite y a la Serie temporal de las Mareas de Venecia. En todos los casos, se ha utilizado como valor de redondeo el valor $c = 1$, que equivale a tomar $n_k = int(f_{nk})$. Este valor sólo tiene influencia

sobre los patrones cuyo valor f_{nk} es ligeramente inferior a 1, por esta razón, se ha decidido utilizar el valor anterior. Además, se han utilizado en todos los casos los siguientes parámetros de entrenamiento de las redes: 500 ciclos de entrenamiento y tasa de aprendizaje $\eta = 0.2$.

Función Definida por Partes

En este dominio se ha aplicado el método descrito anteriormente utilizando redes de 2, 3, ..., 9 neuronas, obteniéndose los resultados mostrados en la tabla 4.11. Se ha decidido no aumentar más el número de neuronas por el incremento del error obtenido. En la figura 4.18 se pueden ver los resultados obtenidos representados gráficamente. Se observa que a partir de 2 neuronas el error decrece, alcanza un mínimo cuando la red tiene 4 neuronas y luego aumenta de nuevo.

Neuronas	ErrMedio
2	0.03998
3	0.01821
4	0.01608
5	0.01888
6	0.02429
7	0.03140
8	0.03688
9	0.05439

Tabla 4.11:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Función Definida por Partes

Cuando la red tiene 2 neuronas se ha obtenido un error relativamente alto, debido a que son insuficientes neuronas para cubrir el espacio de entrada formado por los patrones seleccionados. Cuando el número de neuronas está comprendido entre 3 y 5, es suficiente para cubrir el espacio de entrada formado por los patrones seleccionados y la red consigue un error de generalización pequeño. Si el número de neuronas crece, entonces es demasiado grande para los pocos patrones seleccionados, y el error de generalización aumenta.

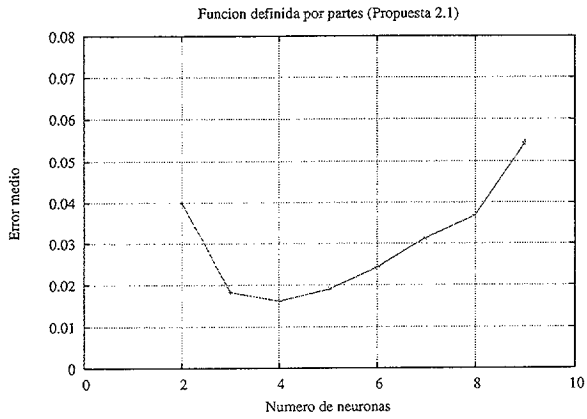


Figura 4.18:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Función Definida por Partes

En la tabla 4.12 se comparan los mejores resultados obtenidos cuando se utiliza el entrenamiento tradicional y cuando se utilizan la ponderación gaussiana (llamada propuesta 1) y el método actual con ponderación inversa (llamado propuesta 2.1), cuando se aplican al dominio de la Función Definida por Partes. Una importante mejora del método es que, siempre que el número de neuronas utilizado sea pequeño (menor que 8), se obtienen mejores resultados que con el entrenamiento tradicional.

Entrenam. selectivo Pond. inversa (prop. 2.1)	Entrenam. selectivo Pond. gaussiana (prop. 1)	Entrenam. tradicional
0.016087 4 neuronas	0.0233 $\sigma = 0.25$, 9 neuronas	0.04156 100 neuronas

Tabla 4.12:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1 y 2.1) y con entrenamiento tradicional. Función Definida por Partes

Polinomio de Hermite

Se ha aplicado el método correspondiente a la propuesta 2.1 al Polinomio de Hermite, habiendo utilizando redes de 2, 4, ..., 24 neuronas, obteniéndose los resultados mostrados en la tabla 4.13, donde pueden verse los errores

medios obtenidos para todo el conjunto de test con diferente número de neuronas.

Neuronas	ErrMedio
2	0.02157
4	0.01948
6	0.01212
8	0.01017
10	0.00571
12	0.00771
14	0.00761
16	0.00779
18	0.00916
20	0.01159
22	0.01678
24	0.06505

Tabla 4.13:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Polinomio de Hermite

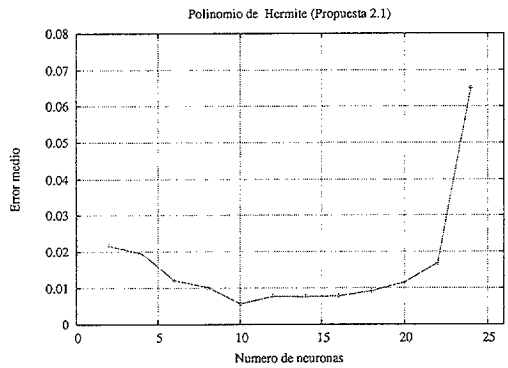


Figura 4.19:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Polinomio de Hermite

En la figura 4.19 se representan gráficamente dichos resultados.

Analizando estos datos se observa que el error es relativamente alto con 2 neuronas, alcanza un mínimo con 10 neuronas, se mantiene cercano a 0.008 entre 12 y 16 neuronas y crece más rápidamente cuando el número de neuronas aumenta. Este crecimiento del error cuando el número de neuronas

es mayor de 20 se debe a que son demasiadas neuronas para un espacio de entrada que contiene muy pocos patrones, provocando un error de generalización muy alto. La tendencia de la curva del error es similar a la de la Función Definida por Partes, aunque el rango de neuronas utilizado es mayor.

En la tabla 4.14 pueden verse los mejores resultados obtenidos por el entrenamiento tradicional y por los métodos correspondientes a las propuestas 1 y 2.1 cuando se aplican al Polinomio de Hermite. La mejora obtenida por el método 2.1 sobre el entrenamiento tradicional es sustancial, pues se obtiene un error medio de 0.00571 frente a un error de 0.01904 cuando se aplica el método convencional, es decir el error es 3.33 veces menor, siendo necesarias 10 neuronas frente a 40.

Entrenam. selectivo Pond. inversa (prop. 2.1)	Entrenam. selectivo Pond. gaussiana (prop. 1)	Entrenam. tradicional
0.00571	0.0121	0.01904
10 neuronas	$\sigma = 0.1$, 5 neuronas	40 neuronas

Tabla 4.14:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1 y 2.1) y con entrenamiento tradicional. Polinomio de Hermite

Serie temporal de las Mareas de Venecia

A continuación se muestran los resultados de aplicar el método propuesto a la serie temporal de las Mareas de Venecia. En los experimentos realizados, se han utilizando redes de 5, 10, ..., 45 neuronas, habiéndose obtenido los resultados mostrados en la tabla 4.15.

En la figura 4.20 se pueden ver los resultados obtenidos representados gráficamente. Se observa un comportamiento similar al de los dominios anteriores, mejorándose el comportamiento de la red respecto del entrenamiento clásico. El error es alto con 5 neuronas, alcanza un mínimo con 20 neuronas, y sube, de forma más suave que en los dominios anteriores, cuando el número de neuronas aumenta.

Neuronas	ErrMedio
5	0.11696
10	0.05820
15	0.04782
20	0.04117
25	0.04559
30	0.04905
35	0.05082
40	0.05318
45	0.05802

Tabla 4.15:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Serie temporal de las Mareas de Venecia

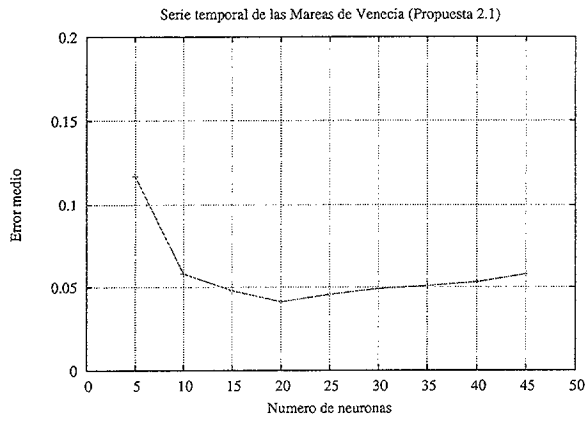


Figura 4.20:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Serie temporal de las Mareas de Venecia

Como se ha hecho en los dominios anteriores, se muestra una tabla (ver tabla 4.16) con los mejores resultados obtenidos para este dominio con el entrenamiento tradicional y con los métodos propuestos hasta el momento. Puede verse que con la ponderación inversa (propuesta 2.1) se mejoran los resultados que se habían obtenido con la ponderación gaussiana (propuesta 1), que a su vez mejoraba ligeramente los resultados obtenidos cuando se entrenaban las RNBR por el método clásico. Además, el número de neuronas sigue siendo bastante menor que el que necesita cuando se entrena la red por el procedimiento tradicional.

Entrenam. selectivo Pond. inversa (prop. 2.1)	Entrenam. selectivo Pond. gaussiana (prop. 1)	Entrenam. tradicional
0.04117 20 neuronas	0.09053 $\sigma = 0.15$, 17 neuronas	0.09605 50 neuronas

Tabla 4.16:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1 y 2.1) y con entrenamiento tradicional. Serie temporal de las Mareas de Venecia

Serie temporal de Mackey-Glass

Por último, se aplica el método descrito en la propuesta 2.1 al dominio de la serie temporal de Mackey-Glass, que es similar la de las Mareas de Venecia en cuanto a la alta dimensionalidad y la irregular distribución de los patrones en el espacio de entrada. Se han utilizando RNBR de 4, 8, 12, ..., 52 neuronas, habiéndose obtenido los errores medios mostrados en la tabla 4.17.

Neuronas	ErrMedio
4	0.08906
8	0.05803
12	0.03815
16	0.02898
20	0.02515
24	0.02511
28	0.02476
32	0.02429
36	0.02650
40	0.02765
44	0.03148
48	0.03942
52	0.04623

Tabla 4.17:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Serie temporal de Mackey-Glass

En la figura 4.21 puede verse la representación gráfica de los datos anteriores. Se observa que el error es grande cuando el número de neuronas es pequeño, se mantiene cercano a 0.024 cuando el número de neuronas está comprendido entre 20 y 36 neuronas, y vuelve a crecer a partir de ese valor. Puede observarse que el error evoluciona de la misma forma que en el resto

de los dominios, aunque el número de neuronas sea diferente. Al ser un dominio más complejo, donde se seleccionan más patrones de entrenamiento, son necesarias más neuronas para realizar una buena generalización.

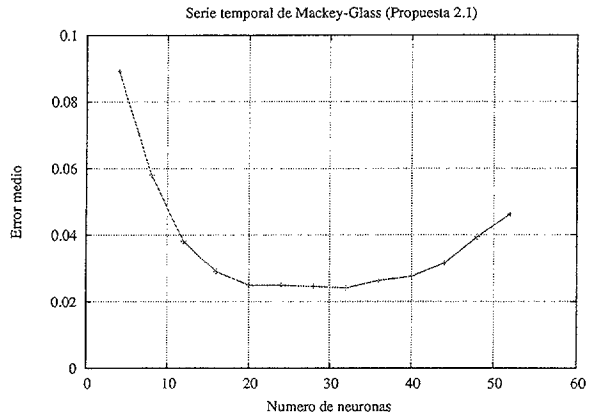


Figura 4.21:
Errores medios con aprendizaje selectivo (Propuesta 2.1). Serie temporal de Mackey-Glass

A continuación se muestran en una tabla comparativa (tabla 4.18) los mejores resultados obtenidos en el dominio de Mackey-Glass cuando se han utilizado RNBR entrenadas por el método tradicional, y cuando se han aplicado los métodos de aprendizaje selectivo correspondientes a las propuestas 1, donde se utiliza la función gaussiana, y a la propuesta 2.1, donde se utiliza la función inversa.

Entrenam. selectivo Pond. inversa(fase 2.1)	Entrenam. selectivo Pond. gaussiana (fase 1)	Entrenam. tradicional
0.02429 32 neuronas	0.0348 $\sigma = 0.1$, 9 neuronas	0.10273 110 neuronas

Tabla 4.18:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1 y 2.1) y con entrenamiento tradicional. Serie temporal de Mackey-Glass

Puede observarse que el método 2.1 mejora ligermente los resultados del método 1, que a su vez mejoraba los resultados del método tradicional. Aunque el número de neuronas necesario para obtener los mejores resultados

con el método 2.1 es mayor que el necesario en el método 1, es mucho menor que el número de neuronas necesario cuando se entrenan las RNBR por el procedimiento clásico.

Conclusiones

Del análisis de los resultados obtenidos con los cuatro dominios utilizados, se han extraído las siguientes conclusiones:

- Los resultados obtenidos por la propuesta 2.1, correspondiente a la ponderación inversa, para los dominios de la Función Definida por Partes, Polinomio de Hermite y las series temporales de las Mareas de Venecia y de Mackey-Glass son mejores que los obtenidos en el método 1, donde se utilizaba la función gaussiana como función kernel para seleccionar los patrones de entrenamiento.

- El método propuesto mejora los resultados obtenidos por el método tradicional con un amplio rango de arquitecturas. Por ejemplo, en el Polinomio de Hermite, el error medio obtenido está por debajo de 0.019 para todas las arquitecturas comprendidas entre 4 y 22 neuronas, mientras que ninguna arquitectura consigue un error inferior a dicho valor cuando se utiliza el método tradicional. Esto ocurre en mayor o menor grado con el resto de los dominios, y hace que el método sea robusto, pues tiene cierto grado de independencia de la arquitectura, haciendo que no sea crítico el número de neuronas. Esta robustez no se daba en método 1, donde había una gran dependencia de la desviación de la gaussiana.

- El **valor de Redondeo** no tiene demasiada influencia sobre la selección de los patrones, puesto que sólo tiene relevancia cuando los valores f_{nk} de los patrones son menores que 1. En este caso, dependiendo del valor de c el valor n_k será 1, en cuyo caso será seleccionado, ó 0 y será descartado. Con los patrones cuyo valor f_{nk} es mayor que 1, la influencia del parámetro c es muy pequeña, pues estos patrones siempre serán seleccionados. Por ejemplo, si $c = 0.7$ y $f_{nk} = 3.8$, n_k será 4, y si el valor de redondeo se hiciera mayor, $c = 0.9$, entonces n_k sería 3. Independientemente del valor del corte, n_k no podría tomar un valor distinto de 3 ó 4. Sería conveniente tener control efectivo sobre los patrones que se desea seleccionar.

- Se han observado dos aspectos importantes relacionados con el algoritmo **K-medias** utilizado para la determinación de los centros de las

neuronas en el entrenamiento de las RNBR. El primer aspecto se refiere a los patrones de entrenamiento utilizados por el algoritmo: en las propuestas 1 y 2.1 se ha utilizado el conjunto de patrones seleccionados sin repetir, de forma que todos los patrones seleccionados tienen la misma influencia en la determinación de los centros de las neuronas. Parece adecuado utilizar el conjunto de patrones repetidos, de forma que los patrones más cercanos al de test tengan más influencia en la determinación de los centros.

El segundo aspecto se refiere a la inicialización del algoritmo K-medias: en las propuestas 1 y 2.1 se ha inicializado según la versión clásica del algoritmo, es decir, si se utilizan k neuronas, se asignarán valores aleatorios dentro del espacio de entrada a los k centros, que irán evolucionando para situarse en los centroides de las k clases en que se dividirá el espacio de entrada. Se ha observado que, en ocasiones, hay clases que quedan vacías, especialmente cuando k es grande. Esto es un inconveniente porque las neuronas cuyos centros corresponden a clases vacías estarán alejadas de casi todos los patrones y su activación, aunque pequeña, perjudicará al ajuste de la red.

4.4.2 Parámetro *corte*. Modificación del algoritmo K-medias

En esta nueva propuesta del método de aprendizaje selectivo con ponderación inversa, que se denominará *propuesta 2.2*, se introducen algunas modificaciones al método descrito en el apartado anterior con el fin de solucionar los inconvenientes citados en las conclusiones anteriores. Estas modificaciones son las siguientes:

- Se elimina el *valor de redondeo*, y en su lugar se introduce un parámetro llamado *corte* representado por el símbolo r, que actuará como un valor umbral de distancia, de forma que los patrones cuya distancia al patrón de test sea mayor que el corte, no serán seleccionados, y el resto sí. De este modo, se establece una hiperesfera de radio r alrededor del patrón de test, de forma que todos los patrones de entrenamiento que estén situados en el interior de esta hiperesfera serán seleccionados, y el resto serán descartados. Se observa que si el corte, y por tanto las distancias, se dan en términos absolutos, se necesitará conocer la magnitud de éstas, para saber si un determinado corte implica la selección de muchos o pocos patrones de entrenamiento. Por tanto, se introduce una nueva modificación en el método transformando el corte y las distancias absolutas en magnitudes relativas respecto del patrón más alejado del patrón de test. El corte r se transforma en un *corte relativo*, r_r, de forma que cuando $r_r = 1$ el radio de la hiperesfera coincidirá con la distancia del patrón de test al patrón de entrenamiento más alejado, seleccionándose, por tanto, todos los patrones de entrenamiento. Asimismo, las distancias absolutas d_k se transformarán en distancias relativas d_{rk}.

- Para solucionar el primer inconveniente citado en las conclusiones de la propuesta 2.1 referente a K-medias, que estaba relacionado con la naturaleza del conjunto de entrenamiento al que se aplicaba el algoritmo, se modifica el método del siguiente modo: En lugar de aplicar el algoritmo K-medias al conjunto que contiene los patrones seleccionados sin repetir, se aplica al conjunto X_q, donde los patrones estarán repetidos tantas veces como indique el número entero asociado n_k. Así, los patrones más cercanos al patrón de test tendrán más influencia para atraer a los centros en K-medias.

- Finalmente, para resolver el segundo inconveniente referente a K-medias, relacionado con la inicialización de los centros, se modifica el algoritmo K-medias inicializando los centros alrededor del centroide

del conjunto de entrenamiento X_q como se describirá en el siguiente apartado.

Descripción del método

Para determinar los patrones que formarán parte del conjunto X_q (conjunto de entrenamiento asociado al patrón de test $\mathbf{q}$), debe establecerse previamente el parámetro corte relativo, $r_r \in [0, 1]$. Este valor será utilizado para seleccionar los patrones de entrenamiento que estén situados en la hiperesfera de radio r_r centrada en el patrón de test $\mathbf{q}$. Tanto el corte como las distancias de los patrones d_{rk} son relativas respecto del patrón más alejado.

Los pasos a seguir para determinar X_q , son los siguientes:

1. Se asocia a cada patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ un valor real d_k, que se define en términos de la distancia euclídea desde el patrón de test $\mathbf{q}$ a cada patrón de entrenamiento. Esta distancia es la misma que la descrita en la propuesta 2.1 (ecuación 4.10).

2. Para cada patrón $(\mathbf{x_k}, \mathbf{y_k})$ se calcula una distancia relativa d_{rk} como:

$$d_{rk} = \frac{d_k}{d_{max}} \tag{4.20}$$

donde

$$d_{max} = \max(d_1, d_2, \ldots, d_N) \tag{4.21}$$

3. Se asocia a cada patrón $(\mathbf{x_k}, \mathbf{y_k})$ el valor resultante de aplicar la función inversa a la distancia relativa d_{rk}:

$$K(\mathbf{x_k}) = \frac{1}{d_{rk}}, k = 1, 2, \ldots, N \tag{4.22}$$

4. Se aplica un factor de normalización V para transformar los valores $K(\mathbf{x_k})$, en otros valores $K_n(\mathbf{x_k}) = VK(\mathbf{x_k})$ de forma que su suma sea igual al número total de patrones de entrenamiento:

$$\sum_{k=1}^{N} K_n(\mathbf{x_k}) = N \tag{4.23}$$

Este valor normalizado se llamará frecuencia relativa f_{nk}, como ocurría en la propuesta 2.1 El factor de normalización V se calcula de la misma forma que en el método descrito en el apartado 4.4.1:

$$V = \frac{N}{\sum_{k=1}^{N} K(\mathbf{x_k})} \tag{4.24}$$

Así,

$$\sum_{k=1}^{N} f_{nk} = N \tag{4.25}$$

5. Los valores f_{nk} previamente calculados se utilizarán para indicar cuántas veces el patrón de entrenamiento seleccionado $(\mathbf{x_k}, \mathbf{y_k})$ será repetido en el nuevo subconjunto de entrenamiento X_q y ,por tanto, deben ser transformados en números naturales. En este caso, la transformación viene dada por:

$$\begin{array}{lll} \text{si} & d_{rk} < r_r & \text{entonces} \\ & n_k = int(f_{nk}) + 1 & \\ \text{si no} & & \\ & n_k = 0 & \end{array} \tag{4.26}$$

   Así, la distancia relativa calculada en el paso 2 se utilizará para decidir si el patrón de entrenamiento $(\mathbf{x_k}, \mathbf{y_k})$ será seleccionado. Si $d_{rk} < r_r$, donde r_r es el corte relativo establecido previamente, entonces el patrón $(\mathbf{x_k}, \mathbf{y_k})$ será incluido en el conjunto de entrenamiento X_q, puesto que se cumplirá que $n_k >= 1$.

   En este momento, cada patrón de entrenamiento en X tiene asociado un número natural n_k que indicará cuántas veces el patrón $(\mathbf{x_k}, \mathbf{y_k})$ será utilizado para entrenar la RNBR cuando el patrón de test sea presentado.

6. Se construye el nuevo conjunto de entrenamiento X_q asociado al patrón de test $\mathbf{q}$. Dado un patrón $(\mathbf{x_k}, \mathbf{y_k})$ perteneciente al conjunto de entrenamiento original X, será incluido en el nuevo conjunto X_q si el valor n_k es mayor que 0. Además, el patrón $(\mathbf{x_k}, \mathbf{y_k})$ será colocado n_k veces de forma aleatoria en el conjunto X_q. Una vez seleccionados los patrones de entrenamiento, la RNBR se entrena con el conjunto X_q.

   Como se mencionó en la sección capítulo 2.1, el entrenamiento de una RNBR supone la determinación de los centros de las neuronas, las anchuras o desviaciones y los pesos. Los centros se calculan en modo no supervisado utilizando el algoritmo K-medias para clasificar el espacio de entrada que estará formado, al contrario que en las propuestas

anteriores, por el conjunto X_q que contiene a los patrones repetidos. Además, para resolver el inconveniente planteado en el último punto de las conclusiones de la propuesta 2.1, se modifica la inicialización de los centros del algoritmo K-medias del siguiente modo:

- Se calcula M_q, el centroide del conjunto de entrenamiento seleccionado X_q.
- Se generan aleatoriamente k centros $(c_{1q}, c_{2q}, \ldots, c_{kq})$ tales que

$$\|c_{jq} - M_q\| < \epsilon, \text{para } j = 1, 2, \ldots, k \tag{4.27}$$

siendo ϵ un número cercano a cero.

De este modo, los centros se van expandiendo partiendo del centro de masas de los patrones, y se puede comprobar que si el número de centros es menor o igual que el de patrones seleccionados no quedan clases vacías. Una vez calculados los centros de las neuronas, se calculan las anchuras o desviaciones como la raíz cuadrada del producto de las distancias a los dos centros más próximos. Finalmente, se calculan los pesos de modo supervisado para minimizar el error cuadrático medio medido en el conjunto de entrenamiento X_q.

Resultados Experimentales

En esta fase, se han realizado dos conjuntos de experimentos: en el primero, llamado *Estudio global de tendencias*, se aplica el método al conjunto completo de test de cada dominio, variando el corte y el número de neuronas de la RNBR, con el objeto de analizar la influencia de estos parámetros en la capacidad de generalización del método, y estudiar cómo han influido las modificaciones adoptadas en esta propuesta del método, en comparación con resultados anteriores. En estos experimentos se muestran las medias de los errores medios medidos sobre 5 simulaciones con diferentes inicializaciones aleatorias del algoritmo K-medias.

En el segundo, llamado *Estudio detallado de ciertos patrones*, se aplica el método a ciertos patrones en particular, para una determinada arquitectura de la red, variando el corte y utilizando 4 inicializaciones distintas de K-medias. El objetivo de estos experimentos es estudiar la influencia de la inicialización de K-medias en la capacidad de generalización de la red. Se hace con patrones determinados en lugar de con el conjunto completo, para que sea más apreciable el efecto de la inicialización de K-medias.

A continuación se muestran los resultados correspondientes al estudio global de tendencias realizado sobre los dominios de la Función Definida por

Partes, el Polinomio de Hermite y las series temporales de las Mareas de Venecia y Mackey-Glass. En todos los casos se han utilizado los mismos valores para la tasa de aprendizaje, $\eta = 0.2$, y el número de ciclos de entrenamiento, 500.

Estudio global de tendencias. Función Definida por Partes

Se ha aplicado el método descrito en la propuesta 2.2 a la Función Definida por Partes, utilizando RNBR con arquitecturas de 3, 7, ..., 27 neuronas, y se ha modificando el corte desde 0.02 hasta 0.3 , aumentándolo a intervalos de 0.04. En la tabla 4.19 se muestran las medias de los errores medios obtenidos sobre todos los patrones del conjunto de test en 5 simulaciones con inicializaciones aleatorias de K-medias diferentes. Como se mencionó anteriormente, el objetivo de estos experimentos es estudiar la influencia del corte relativo o radio relativo de la hiperesfera de selección en la capacidad de generalización de las RNBR para diferentes arquitecturas.

	Neuronas Ocultas						
Corte	3	7	11	15	19	23	27
0.02	0.00984	0.00975	0.01178	0.01643	0.03824	0.03424	0.05668
0.06	0.01328	0.00774	0.00315	0.00289	0.00273	0.00389	0.00469
0.1	0.01983	0.01063	0.00401	0.00332	0.00285	0.00348	0.00315
0.14	0.02918	0.01079	0.00563	0.00440	0.00374	0.00373	0.00316
0.18	0.03532	0.01441	0.00861	0.00567	0.00474	0.00405	0.00366
0.22	0.04075	0.01865	0.01262	0.00848	0.00565	0.00461	0.00420
0.26	0.04602	0.02492	0.01506	0.01130	0.00675	0.00545	0.00477
0.3	0.05072	0.02787	0.02028	0.01340	0.00855	0.00607	0.00508

Tabla 4.19:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Función Definida por Partes

En la figura 4.22 se representan gráficamente los valores de la tabla 4.19. En esta gráfica puede observarse que cuando el corte relativo es menor de 0.06, el error medio para todas las arquitecturas es muy grande, comportándose peor las redes de mayor número de neuronas. Esto es debido al reducido número de patrones que se seleccionan cuando el corte es tan pequeño, insuficiente para permitir un entrenamiento adecuado de la red, siendo tanto peor este comportamiento cuanto mayor número de neuronas tenga la red. Según va aumentando el corte relativo, el comportamiento de la red depende de su arquitectura: si el número de neuronas es muy pequeño - 3 neuronas - el error aumenta a medida que aumenta el corte alcanzando

valores muy grandes. Con 7, 11 y 15 neuronas, el error alcanza un mínimo cuando el valor del corte es 0.08 y vuelve a aumentar a medida que el corte aumenta. Cuando el número de neuronas es mayor - 19 o más neuronas - el error medio disminuye a medida que el corte aumenta, y para valores de éste mayores o iguales a 0.1 el error se mantiene prácticamente constante.

Este comportamiento puede explicarse del siguiente modo: cuando el corte es muy pequeño, el número de patrones seleccionados es muy reducido e insuficiente para realizar una adecuada generalización. Con este número tan reducido de patrones de entrenamiento, el error será tanto mayor cuanto mayor sea el número de neuronas, pudiéndose dar el caso de que haya más neuronas que patrones. A medida que el corte aumenta, más patrones son seleccionados, permitiendo a la red un mejor comportamiento con el conjunto de test. Si el número de neuronas es pequeño, no será suficiente para ajustarse a un elevado número de patrones, con lo que el error será alto, tanto mayor cuanto mayor sea el corte. Si el número de neuronas es mayor -más de 19- la red puede ajustarse al conjunto de entrenamiento incluso cuando el número de patrones es muy grande, manteniéndose el valor del error muy pequeño y prácticamente constante.

El valor mínimo del error alcanzado en esta prueba es de 0.00273, con una red de 19 neuronas y un corte de 0.06.

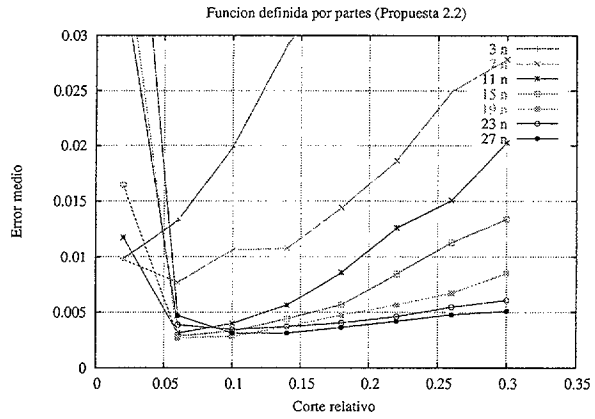


Figura 4.22:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Función Definida por Partes

Estudio global de tendencias. Polinomio de Hermite

También se ha aplicado el método al polinomio de Hermite, utilizando RBN's con arquitecturas de 3, 7, ..., 23 neuronas. Además se ha modificando el corte desde 0.04 hasta 0.28, aumentándolo a intervalos de 0.04. En la tabla 4.20 se muestran las medias de los errores medios obtenidos sobre todos los patrones del conjunto de test en 5 simulaciones con inicializaciones aleatorias de K-medias diferentes.

En la figura 4.23 se representan gráficamente los datos de la tabla 4.20 observándose unas tendencias similares a las del dominio anterior. El error es muy grande cuando el corte es muy pequeño, disminuye mucho cuando el corte alcanza un valor cercano a 0.05, y si la red tiene un número suficiente de neuronas se mantiene prácticamente constante aunque el corte siga aumentando. Como en el caso de la Función Definida por Partes, cuando el corte es muy pequeño el reducido número de patrones seleccionados es insuficiente para una adecuada generalización. A medida que aumenta el corte y se selecciona un mayor número de patrones, aumenta la capacidad de generalización de las redes, aunque las arquitecturas de pocas neuronas no se ajustan adecuadamente al creciente número de patrones produciendo un error que aumenta con el corte. Sin embargo, si el número de neuronas es suficientemente grande - más de 11 neuronas - el error disminuye al principio y luego se mantiene prácticamente constante ya que la red se ajusta de forma adecuada al número de patrones seleccionados.

	Neuronas Ocultas					
Corte	3	7	11	15	19	23
0.04	0.03321	0.02755	0.04067	0.07421	0.11557	0.68453
0.08	0.01037	0.00701	0.00345	0.00999	0.01792	0.02219
0.12	0.01498	0.01112	0.00439	0.00395	0.01093	0.00901
0.16	0.02176	0.01645	0.00635	0.00413	0.01012	0.00561
0.2	0.02902	0.01854	0.00703	0.00502	0.00421	0.00526
0.24	0.03425	0.02204	0.01065	0.00687	0.00488	0.00464
0.28	0.04230	0.02777	0.01450	0.00847	0.00493	0.00547

Tabla 4.20:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Polinomio de Hermite

Se alcanza el valor mínimo del error, 0.00345, con una red de 11 neuronas y con un corte relativo de 0.08, es decir, cuando el radio de la hiperesfera centrada en el correspondiente patrón de test es el 8 % de la distancia de ese patrón de test al patrón de entrenamiento más alejado.

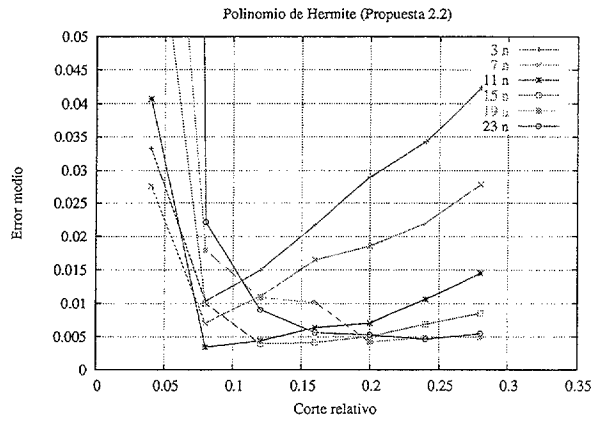


Figura 4.23:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Polinomio de Hermite

Estudio global de tendencias. Serie temporal de las mareas de Venecia

A continuación se aplica el método 2.2 a la serie temporal de las mareas de Venecia, utilizando RNBR con arquitecturas de 3, 7, ..., 27 neuronas, utilizando como corte relativo los valores de 0.04 hasta 0.2, aumentándolo a intervalos de 0.04. En la tabla 4.21 se muestran las medias de los resultados obtenidos en las 5 simulaciones.

	Neuronas Ocultas						
Corte	3	7	11	15	19	23	27
0.04	0.17834	0.28340	0.41402	0.54445	0.75745	0.80031	0.88016
0.08	0.08007	0.06792	0.09453	0.11243	0.10665	0.13414	0.15672
0.12	0.07184	0.03751	0.03452	0.03564	0.02888	0.03088	0.03271
0.16	0.07398	0.03531	0.03805	0.03577	0.03437	0.03406	0.03053
0.2	0.09728	0.04453	0.03954	0.03213	0.03226	0.03692	0.03383

Tabla 4.21:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Serie temporal de las Mareas de Venecia

En la figura 4.24 puede verse la representación gráfica de los resultados anteriores. Cabe destacar el alto error que se produce cuando los cortes son inferiores a 0.1. Esto es debido a que se seleccionan muy pocos patrones, insuficientes para que la red pueda generalizar adecuadamente.

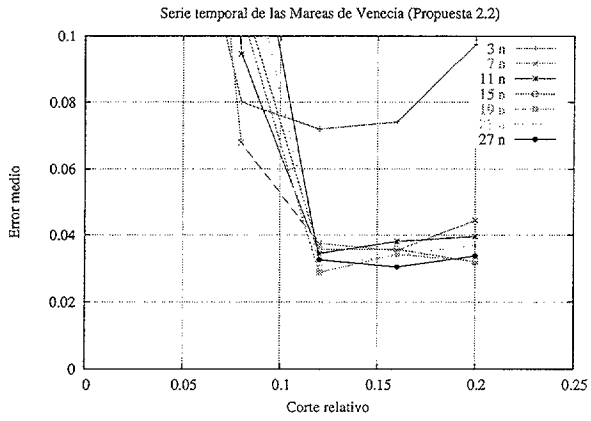


Figura 4.24:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Serie temporal de las Mareas de Venecia

En el conjunto de test existen patrones especialmente "difíciles", son los que representan las situaciones de marea alta, que al ser muy infrecuentes, tienen muy pocos patrones similares en el conjunto de entrenamiento. Por eso, es posible que al ser el corte muy pequeño ocurra que no se seleccione ningún patrón de entrenamiento para estos patrones de test. En estos casos, la red producirá salidas totalmente arbitrarias, elevando considerablemente el error medio. Cuando el corte aumenta, se selecciona un número mayor de patrones de forma que el error disminuye considerablemente. De la misma forma que en los dominios anteriores, las arquitecturas con menor número de neuronas se comportan peor a medida que el corte crece, por las mismas razones explicadas previamente. Es importante destacar el hecho de que si el número de neuronas es suficientemente alto, (mayor que 9), el error se mantiene casi constante cuando aumenta el corte, puesto que las redes pueden ajustarse adecuadamente al número de patrones seleccionado. Como puede verse en la tabla 4.21, el valor mínimo del error conseguido en estos experimentos es de 0.02888 con una red de 19 neuronas y un corte de 0.12.

Estudio global de tendencias. Serie temporal de Mackey-Glass

Por último, se ha aplicado este método a la serie temporal de Mackey-Glass, utilizando RNBR con arquitecturas de 5, 10, ..., 30 neuronas . Se ha modificando el corte desde 0.04 hasta 0.24, aumentándolo a intervalos de 0.04. En la tabla 4.22 se muestran los resultados obtenidos.

	Neuronas Ocultas					
Corte	5	10	15	20	25	30
0.04	0.19019	0.35263	0.47363	0.61413	0.71722	0.85898
0.08	0.06357	0.08142	0.11313	0.12392	0.16735	0.16419
0.12	0.03749	0.02653	0.02171	0.02104	0.02005	0.01877
0.16	0.04561	0.02337	0.01721	0.01863	0.01651	0.01711
0.2	0.05814	0.02041	0.01996	0.01691	0.01722	0.01713
0.24	0.06658	0.02265	0.02076	0.01824	0.01873	0.01739

Tabla 4.22:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Serie temporal de Mackey-Glass

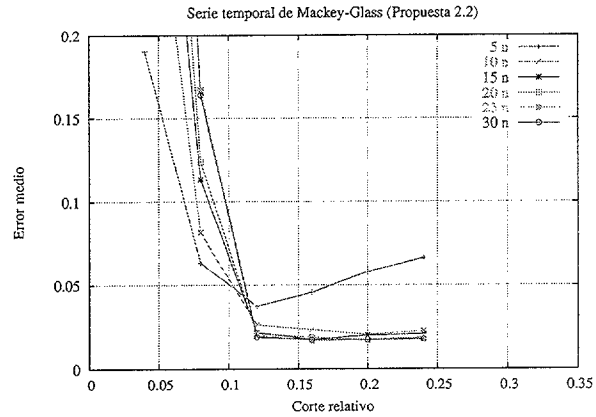


Figura 4.25:
Errores medios con aprendizaje selectivo (Propuesta 2.2). Serie temporal de Mackey-Glass

En la figura 4.25 puede verse la representación gráfica de los resultados anteriores. También en este caso se produce un error muy grande cuando los cortes son inferiores a 0.1. Esto también es debido a que se seleccionan muy pocos patrones, insuficientes para que la red pueda generalizar adecuadamente. En el conjunto de test, al igual que en las mareas de Venecia, existen patrones especialmente "difíciles", que al ser muy infrecuentes, tienen muy pocos patrones similares en el conjunto de entrenamiento. Por eso, también puede ocurrir que no se seleccione ningún patrón de entrenamiento cuando los cortes son muy pequeños, produciendo la red salidas arbitrarias, que eleven considerablemente el error medio. El valor mínimo del error conseguido en esta prueba es de 0.01651 con una red de 25 neuronas

y un corte de 0.16. Al igual que en los dominios anteriores se observa que cuando el número de neuronas es suficientemente alto, el error se mantiene prácticamente constante cuando aumenta el corte.

Estudio global de tendencias. Comparación de resultados

En la tabla 4.23 se comparan los mejores resultados obtenidos en la propuesta 2.2 con los obtenidos en las propuestas anteriores y con el método tradicional, para los dominios estudiados. Puede verse que los mejores errores obtenidos en esta propuesta, son menores que los obtenidos en cualquiera de las propuestas anteriores.

Error Medio	Propuesta 2.2	Propuesta 2.1	Propuesta 1	Entrenam. tradicional
Función definida por partes	0.00273 $r_r = 0.06$, 19 n	0.016087 4 neuronas	0.0233 σ=0.25, 9 n	0.04156 100 n
Polinomio de Hermite	0.00345 $r_r = 0.08$, 9 n	0.00571 10 n	0.0121 σ=0.1, 5 n	0.01904 40 n
Serie de las Mareas de Venecia	0.02888 $r_r = 0.12$, 19 n	0.04117 20 n	0.0905 σ=0.15, 17 n	0.0605 50 n
Serie de Mackey-Glass	0.01651 $r_r = 0.15$, 25 n	0.02429 32 n	0.0348 σ=0.1, 9 n	0.10273 110 n

Tabla 4.23:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1, 2.1 y 2.2) y con entrenamiento tradicional

Estudio detallado de ciertos patrones. Función Definida por Partes

En este apartado se trata de estudiar la influencia que tienen las inicializaciones aleatorias de los centros en el algoritmo K-medias. Se ha decidido hacer un estudio detallado de ciertos patrones de test aislados para resaltar las diferencias producidas en las distintas inicializaciones aleatorias. Se ha realizado una prueba preliminar, utilizando una patrón especialmente complicado de generalizar, y es el que corresponde al "pico" de la Función Definida por Partes (ver figura 4.1), donde se produce un cambio de tendencia. Se fija el número de neuronas en 31, puesto que en la tabla 4.19 se vio que con ese número de neuronas el corte tenía poca influencia en el error, y se varía el corte entre 0.05 y 0.5 con incrementos de 0.05. Se realizan los experimentos con 4 inicializaciones aleatorias diferentes de los centros de K-medias. En la tabla 4.24 se muestran los errores obtenidos para cada

corte en cada una de las 4 inicializaciones, así como la media de los 4 errores y la desviación estándar. También se han representado gráficamente los datos en la figura 4.26, donde puede apreciarse la gran influecia que tiene la inicialización en el error de generalización del patrón de test.

Corte	Aleat 1	Aleat 2	Aleat 3	Aleat 4	Err Medio	SDev
0.05	0.03838	0.00243	0.03178	0.02035	0.02323	0.01574
0.1	0.04874	0.01754	0.04367	0.03433	0.03607	0.01372
0.15	0.04386	0.03358	0.03872	0.04212	0.03957	0.00453
0.2	0.06095	0.05688	0.05393	0.06205	0.05845	0.00375
0.25	0.07514	0.05126	0.05964	0.05634	0.06059	0.01029
0.3	0.08565	0.04291	0.0571	0.05306	0.05968	0.01831
0.35	0.08166	0.03582	0.04843	0.04928	0.0538	0.01957
0.4	0.06869	0.03292	0.04151	0.03253	0.04391	0.01703
0.45	0.07209	0.03907	0.03539	0.03428	0.04521	0.01804
0.5	0.06063	0.04382	0.04976	0.05159	0.05145	0.00696

Tabla 4.24:
Errores obtenidos con el patrón número 32 para las distintas inicializaciones de K-medias (Propuesta 2.2). Función Definida por Partes

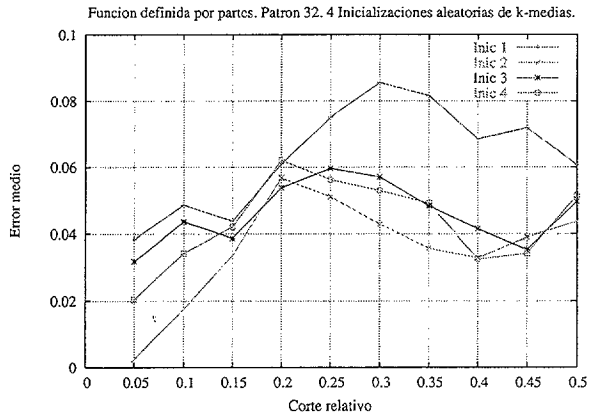


Figura 4.26:
Errores obtenidos con el patrón número 32 para las distintas inicializaciones de K-medias (Propuesta 2.2). Función Definida por Partes

Conclusiones

A continuación se exponen las conclusiones extraídas al analizar los datos obtenidos en esta propuesta del método.

- **El corte no es un parámetro crítico.** Una importante conclusión de la experimentación realizada es que, en esta propuesta, el corte no es un parámetro crítico, ya que a partir de un valor lo suficientemente grande para que se seleccione un número mínimo de patrones, el error no varía sustancialmente, siempre que la red tenga un número suficiente de neuronas. Lógicamente, si el corte es muy pequeño, se selecciona un número de patrones insuficiente para poder generalizar adecuadamente. Si el corte sobrepasa un cierto valor, el número de patrones seleccionados es suficiente para poder generalizar bien, de forma que si se sigue aumentando el corte, el empeoramiento en la capacidad de generalización de la red no es significativo, teniendo un número suficiente de neuronas. Si el número de neuronas es pequeño, la red puede ajustarse bien a los pocos patrones de entrenamiento seleccionados con el corte pequeño, pero si aumenta este número, la red comenzará a tener dificultades para ajustarse a tantos patrones de entrenamiento.

- Los resultados obtenidos en esta propuesta mejoran significativamente los obtenidos en las propuestas anteriores, y los obtenidos cuando se entrenan las redes por el método convencional. Además, el método es robusto pues es relativamente independiente de la arquitectura de las redes y del corte elegido, siempre que estos parámetros estén dentro de un rango relativamente amplio.

- **Resultados muy dependientes de las inicializaciones aleatorias de K-medias.** Esta es otra conclusión importante que se extrae de los resultados experimentales. El algoritmo K-medias, utilizado para situar los centros de las neuronas, alcanza mínimos locales que dependen de la inicialización de los centros. La capacidad de generalización de las redes de base radial depende mucho de la correcta situación de los centros de las neuronas y, por tanto, dependerá mucho de la inicialización aleatoria de K-medias. Se ha visto en los experimentos con patrones concretos, que el error alcanzado en la generalización depende mucho de la inicialización de los centroides de K-medias.

4.4.3 Inicialización determinista de K-medias.

En la propuesta 2.2 del método, se observa que los resultados dependen en gran medida de la inicialización aleatoria de los centros de K-medias. Por este motivo, se presenta una nueva propuesta, que se denominará *propuesta 2.3*, donde se pretende evitar esta dependencia de las inicializaciones aleatorias modificando el método para hacer que esta inicialización sea determinista, inicializando los centros a las posiciones de los patrones de entrenamiento seleccionados más repetidos. El método solamente varía en la inicialización del algoritmo K-medias, tal como se describe a continuación:

- Sean $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_l})$ los l patrones de entrenamiento seleccionados para responder al patrón de test $\mathbf{q}$, ordenados según los valores de sus frecuencias normalizadas asociadas $(f_{n1}, f_{n2}, \ldots, f_{nl})$.
- Sea m el número de neuronas de la RNBR que se va a entrenar.
- Si $m \leq l$, entonces el centro de la neurona i se inicializará a la posición de $\mathbf{x_i}$, para $i = 1, 2, \ldots, m$, es decir se inicializarán a las posiciones de los m primeros patrones.
- Si $m > l$, entonces las l primeras neuronas se inicializarán según lo dicho en el punto anterior, y las $m - l$ restantes, se inicializarán aleatoriamente según el método descrito en la propuesta 2.2. Este es un caso poco frecuente, pues casi siempre es mayor el número de patrones seleccionados que el número de neuronas, pero aunque ocurra que $m > l$, los grados de libertad disminuyen mucho, evitando la variabilidad de resultados observados en la propuesta 2.2.

Resultados Experimentales

Se va a aplicar el método, modificado con esta inicialización determinista, a los dominios utilizados en las fases anteriores, y además se aplicará al dominio de clasificación "Pima-indians diabetes", descrito en el apartado 4.1.3. Debido a la alta dimensionalidad de las mareas de Venecia y de Mackey-Glass y del dominio Pima-Indians Diabetes (8 dimensiones), se han querido estudiar con detalle las situaciones donde el corte relativo es pequeño, para detectar los casos donde para ciertos patrones de test no se selecciona ningún patrón de entrenamiento cuando los cortes son pequeños, debido a que en los espacios de alta dimensionalidad las distancias son más grandes y las densidades de patrones son más bajas (el espacio está más "vacío"). Por este motivo, en las tablas que se muestran en esta propuesta se ha

añadido una columna que se ha llamado *'Ceros'* que corresponde al número de patrones de test para los cuales no se ha seleccionado ningún patrón de entrenamiento.

A continuación se muestran los resultados obtenidos al aplicar el método 2.3 a los cinco dominios. En todos los casos se han utilizado los valores de la tasa de aprendizaje y número de ciclos que se habían utilizado en las propuestas anteriores.

Función Definida por Partes

Aplicando el método a la Función Definida por Partes, se obtienen los resultados mostrados en la tabla 4.25. Se han utilizado redes de 3, 7, ..., 31 neuronas. El corte varía desde 0.02 hasta 0.3 con un incremento de 0.04.

	Neuronas Ocultas							
Corte	3	7	11	15	19	23	27	Ceros
0.02	0.00995	0.00883	0.00927	0.01027	0.01707	0.03594	0.06726	0
0.06	0.01418	0.00690	0.00208	0.00257	0.00255	0.00237	0.00785	0
0.1	0.02061	0.00872	0.00407	0.00519	0.00302	0.00237	0.00246	0
0.14	0.03051	0.00938	0.00719	0.00583	0.00391	0.00516	0.00404	0
0.18	0.03760	0.01515	0.00816	0.00459	0.00351	0.00375	0.00507	0
0.22	0.03698	0.02060	0.01266	0.00914	0.00398	0.00415	0.00393	0
0.26	0.04620	0.02550	0.01760	0.01059	0.00541	0.00518	0.00505	0
0.3	0.04917	0.03061	0.01937	0.01411	0.00660	0.00659	0.00640	0

Tabla 4.25:
Errores medios con aprendizaje selectivo (Propuesta 2.3). Función Definida por Partes

Se observa que se alcanzan resultados ligeramente mejores que en la propuesta 2.2. En concreto, cabe destacar el error de 0.00208 para un corte relativo de 0.06 y una red de 11 neuronas. En la figura 4.27 puede verse la representación gráfica de los datos anteriores. Al igual que en la propuesta 2.2, el error se mantiene prácticamente constante para cortes superiores a 0.1, siempre que la red tenga un número suficiente de neuronas (más de 19). También se observa que con cortes muy pequeños el error depende mucho del número de neuronas. Si el número de neuronas es pequeño, la red se adapta relativamente bien al escaso número de patrones seleccionados, pero si el número de neuronas es grande, la red generaliza muy mal ya que se ajustan una gran cantidad de parámetros de la red para adaptarse a muy pocos patrones de entrenamiento; esto hace que una función compleja se adapte muy bien a estos patrones de entrenamiento, pero se producirá un

mayor error de generalización. Al aumentar el corte, el número de patrones seleccionados aumenta, y si la red tiene pocas neuronas, éstas son insuficientes para cubrir el espacio de entrada produciéndose un error creciente con el corte como se puede ver en la gráfica. Sin embargo, si el número de neuronas es suficientemente grande, la red se adapta al número de patrones seleccionados, produciéndose un error de generalización pequeño y prácticamente independiente del corte.

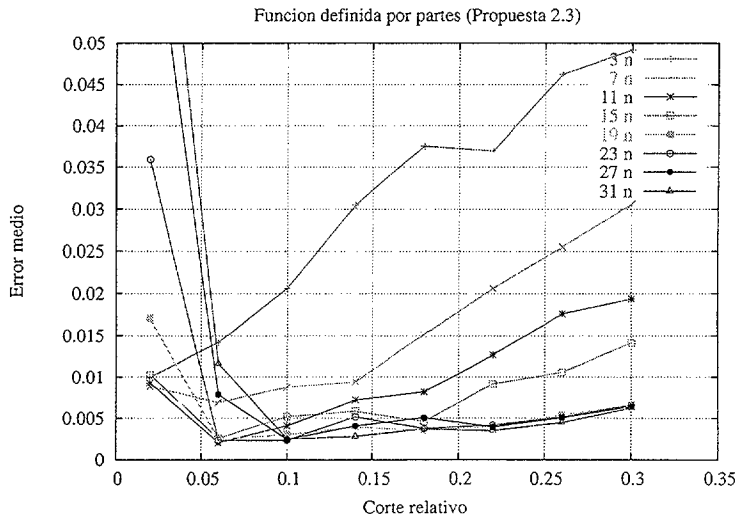


Figura 4.27:
Errores medios con aprendizaje selectivo (Propuesta 2.3). Función Definida por Partes

También es destacable el hecho de que para todos los valores del corte, con todos los patrones de test se seleccionan patrones de entrenamiento, es decir, todos los patrones de test tienen patrones de entrenamiento lo suficientemente cerca para que éstos sean seleccionados, incluso para los cortes más pequeños.

En la figura 4.28 se ha representado el error de generalización cometido para cada uno de los 80 patrones de test de este dominio, cuando se entrena la red por el método convencional utilizando la mejor arquitectura (ver sección 4.2.1), y cuando se utiliza el método selectivo correspondiente a la propuesta 2.3 utilizando los valores de corte y número de neuronas que producen los mejores resultados. Puede verse que el error cometido por el método tradicional para casi todos los patrones es significativamente mayor

que el correspondiente al método selectivo.

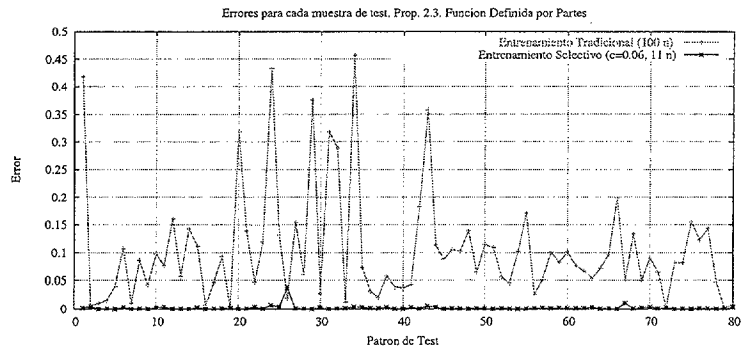


Figura 4.28:
Errores para cada patrón de test (Propuesta 2.3). Función Definida por Partes

Es importante destacar que este método obtiene buenos resultados sin depender de inicializaciones aleatorias.

Polinomio de Hermite

Aplicando el método 2.3 al Polinomio de Hermite se han utilizado arquitecturas de 3, 7, ..., 23 neuronas y cortes relativos desde 0.04 hasta 0.28 con incrementos de 0.04. Se han obtenido los valores de los errores medios que se muestran en la tabla 4.26, los cuales se representan gráficamente en la figura 4.29.

	Neuronas Ocultas						
Corte	3	7	11	15	19	23	Ceros
0.04	0.02057	0.0135	0.0237	0.0621	0.1063	0.1753	0
0.08	0.0112	0.0062	0.0042	0.0060	0.0120	0.0245	0
0.12	0.0170	0.0106	0.0038	0.0058	0.0053	0.0078	0
0.16	0.0225	0.0123	0.0048	0.0040	0.0051	0.0056	0
0.2	0.0296	0.0153	0.0058	0.0047	0.0042	0.0072	0
0.24	0.0353	0.0198	0.0093	0.0064	0.0044	0.0046	0
0.28	0.0436	0.0228	0.0129	0.0064	0.0046	0.0044	0

Tabla 4.26:
Errores medios con aprendizaje selectivo (Propuesta 2.3). Polinomio de Hermite

En la figura 4.29 se ve que los resultados son ligeramente peores que los obtenidos para el mismo dominio en la propuesta 2.2, destacando el error de 0.0038 correspondiente a una red de 11 neuronas y corte 0.12. Lo realmente importante es que ahora estos resultados no dependen de ninguna inicialización aleatoria. Al igual que en el dominio anterior, con un número de neuronas suficientemente grande el error apenas varía cuando aumenta el corte, siempre que éste sea mayor que cierto valor, en este caso 0.12. Igual que en los casos anteriores se observa que con cortes muy pequeños el error es muy dependiente de la arquitectura de la red: para redes con pocas neuronas el error es relativamente pequeño, ya que hay muy pocos patrones de entrenamiento seleccionados, y para redes con gran número de neuronas el error de generalización es muy grande por los mismos motivos que se explicaron en casos anteriores.

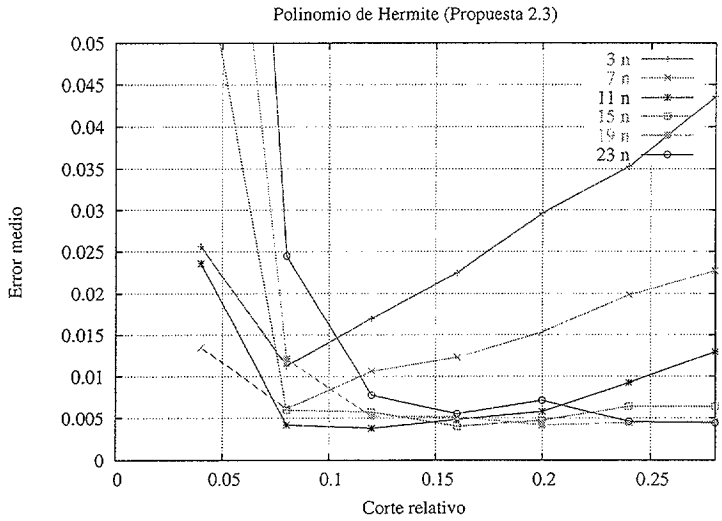


Figura 4.29:
Errores medios con aprendizaje selectivo (Propuesta 2.3). Polinomio de Hermite

Al aumentar el corte y aumentar el número de patrones seleccionados, si la red tiene pocas neuronas, éstas son insuficientes para cubrir el espacio de entrada produciéndose un error creciente con el corte como se puede ver en la gráfica. Sin embargo, si el número de neuronas es suficientemente grande, la red se adapta al número de patrones seleccionados, produciéndose un error de generalización pequeño que se mantiene prácticamente constante

según aumenta el corte. Al ser este dominio unidimensional y estar los datos uniformemente distribuidos en el espacio de entrada, todos los patrones de test tienen datos de entrenamiento lo suficientemente cercanos con lo que siempre se seleccionarán patrones para entrenar las redes. Por este motivo, se ve en la columna 'Ceros' de la tabla 4.26 que no existen patrones de test para los que no se seleccionen patrones de entrenamiento.

En la figura 4.30 pueden verse las representaciones gráficas de los errores de generalización obtenidos para cada patrón de test cuando se utiliza el método tradicional entrenando una RNBR de 40 neuronas, y cuando se utiliza el método de la propuesta 2.3 con una red de 11 neuronas y con un corte relativo de 0.12. Ambas configuraciones son las que obtienen los mejores resultados en sus respectivos métodos de entrenamiento. Puede observarse en las curvas, que para la gran mayoría de los patrones el error de generalización es mucho menor cuando se utiliza el método selectivo 2.3.

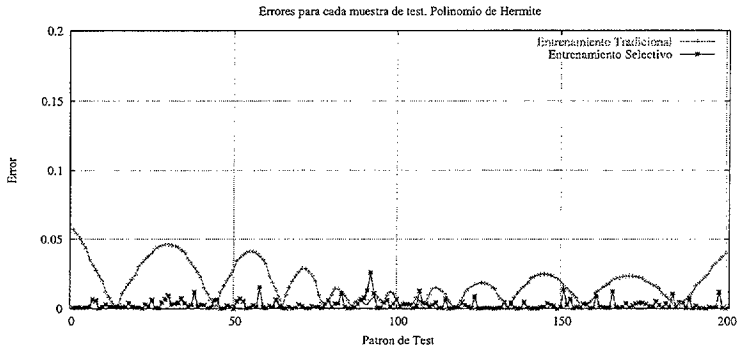


Figura 4.30:
Errores para cada patrón de test (Propuesta 2.3). Polinomio de Hermite

Serie temporal de las Mareas de Venecia

También se ha aplicado el método propuesto al dominio de las mareas de Venecia, con redes de arquitecturas de 3, 7, ... 27 neuronas, aplicando cortes relativos de 0.04 a 0.2 con incrementos de 0.04. Los errores medios obtenidos se muestran en la tabla 4.27.

Éste es un dominio de alta dimensionalidad donde las distancias entre patrones son más grandes y además la distribución de los datos en el espacio de entrada no es uniforme, existiendo zonas de ese espacio con muy baja densidad de datos, como ya se ha comentado anteriormente. Este hecho se manifiesta en los valores que aparecen en la columna 'Ceros' de la tabla:

cuando el corte relativo es 0.04, existen 14 muestras del conjunto de test para los que no se selecciona ningún patrón de entrenamiento. Cuando el valor del corte es 0.08, existen solamente 2 muestras de test en esta situación, y si el corte es 0.12 o mayor entonces ya no existe ninguna muestra de test en esta situación anómala. Esto quiere decir que si el radio de la hiperesfera es menor o igual que el 4% de la distancia al patrón de entrenamiento más alejado y se sitúa centrada en alguno de estos 14 patrones de test , entonces no existirá ninguna muestra de entrenamiento en el interior de esta hiperesfera. Si su radio crece hasta el 8% de dicha distancia máxima ya existirán patrones de entrenamiento en su interior, excepto cuando se centra la hiperesfera en dos patrones de test. Si el radio es mayor o igual que el 12% de la distancia máxima, siempre existirán patrones de entrenamiento en su interior, independientemente del patrón de test que se elija para centrar la hiperesfera.

En esta propuesta, al haberse detectado estas situaciones anómalas, no se ha computado el error producido por las redes para estos patrones en el error medio. Por este motivo aparecen errores bastante pequeños para los cortes de 0.04 y 0.08, pero no son comparables al resto puesto que parte de los patrones de test no se han tenido en cuenta.

	Neuronas Ocultas							
Corte	3	7	11	15	19	23	27	Ceros
0.04	0.03445	0.03330	0.03322	0.03600	0.03534	0.03720	0.03429	14
0.08	0.04434	0.03510	0.03125	0.03145	0.02591	0.03232	0.03198	2
0.12	0.05457	0.02967	0.02682	0.02269	0.02234	0.02235	0.02643	0
0.16	0.07463	0.02869	0.02398	0.02913	0.02059	0.02514	0.02552	0
0.2	0.08459	0.03769	0.02420	0.02411	0.02728	0.02288	0.03336	0

Tabla 4.27:
Error medio con aprendizaje selectivo (Propuesta 2.3). Serie temporal de las Mareas de Venecia

En la figura 4.31 se muestran los errores representados gráficamente en función del corte relativo y para distintas arquitecturas. Llama la atención el bajo error que se produce con cortes pequeños pero no debe ser tenido en cuenta según lo dicho en el párrafo anterior, ya que no se han computado los patrones que producen "Ceros", que son la mayoría - 14 frente a 20 en total -. Es decir, sólo se han computado los errores de 6 patrones, que son precisamente los patrones situados en zonas de alta densidad, por tanto corresponden a situaciones normales de marea.

Hay que destacar el valor 0.02059 para el error, con arquitectura de 19

neuronas y un corte relativo de 0.16. Los resultados son similares a los obtenidos en la fase 2.2, pero ahora no dependen de la inicialización.

También ocurre lo que ocurría en el resto de los dominios en esta propuesta y en la anterior: Cuando el número de neuronas es suficientemente grande -en este caso mayor de 11 neuronas- el error no sufre variaciones de importancia cuando aumenta el corte a partir de 0.12.

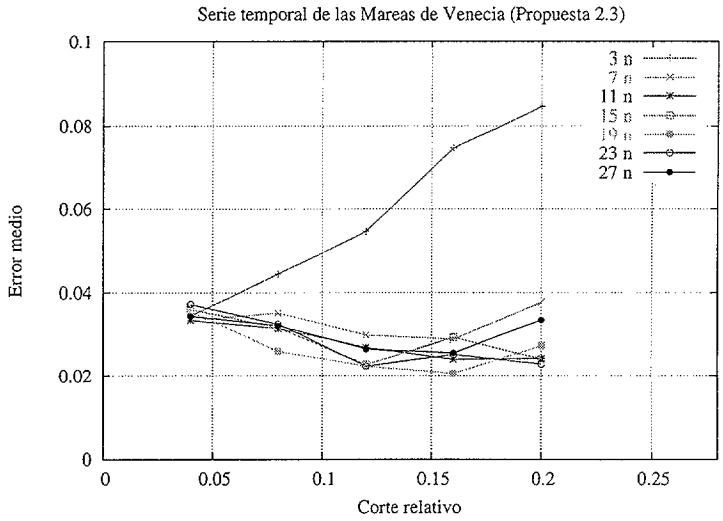


Figura 4.31:
Error medio con aprendizaje selectivo (Propuesta 2.3). Serie temporal de las Mareas de Venecia

Igual que se ha hecho en los dominios anteriores se ha realizado una comparativa de los errores obtenidos para cada patrón de test.

En la figura 4.32 se muestran las gráficas correspondientes a los errores obtenidos para cada uno de los 20 patrones de test cuando se entrena la red - con 50 neuronas - por el procedimiento clásico y cuando se aplica el método 2.3 en cuyo caso se ha utilizado una red de 19 neuronas con un corte relativo de 0.16. Estas configuraciones son las que obtienen los mejores resultados con los respectivos métodos.

Puede observarse en la gráfica 4.32 que los errores de generalización son mucho menores cuando se aplica el método correspondiente a la propuesta 2.3, para casi todos los patrones de test.

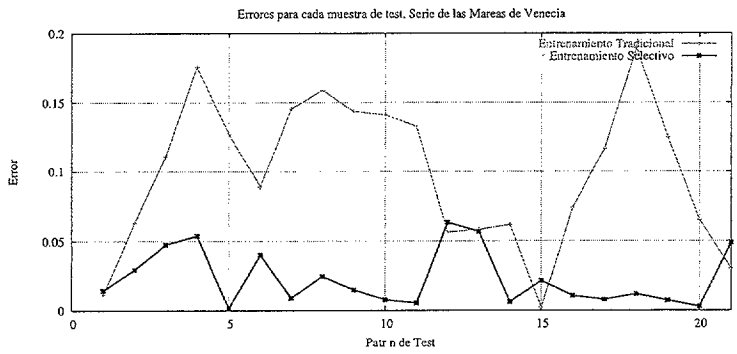


Figura 4.32:
Errores para cada patrón de test (Propuesta 2.3). Serie temporal de las Mareas de Venecia

Serie temporal de Mackey-Glass

Al aplicar la propuesta 2.3 al dominio de la serie temporal de Mackey-Glass, para arquitecturas de 5, 10, ..., 30 neuronas y utilizando cortes relativos de 0.05, 0.1, ..., 0.3, se obtienen los errores medios que se muestran en la tabla 4.28.

	Neuronas Ocultas						
Corte	5	10	15	20	25	30	Ceros
0.04	0.08307	0.08429	0.08567	0.08690	0.08771	0.08852	45
0.08	0.02092	0.01897	0.01666	0.01564	0.01561	0.01554	0
0.12	0.02415	0.01821	0.01644	0.01571	0.01650	0.01711	0
0.16	0.02917	0.02099	0.01812	0.01802	0.01847	0.01936	0
0.2	0.03522	0.02169	0.01927	0.01866	0.02109	0.02156	0
0.24	0.04084	0.02543	0.02255	0.02192	0.02353	0.02472	0

Tabla 4.28:
Error medio con aprendizaje selectivo (Propuesta 2.3). Serie temporal de Mackey-Glass

En la columna "Ceros", puede verse que vuelven a darse las situaciones anómalas explicadas en el apartado anterior, donde para ciertos patrones de test no se selecciona ningún patrón de entrenamiento. En el dominio de Mackey-Glass, los patrones equivalen a puntos en un espacio de 4 dimensiones y no están uniformemente distribuidos en dicho espacio. Con este número de dimensiones, las distancias entre puntos son grandes ya que existen regiones de este espacio donde la densidad de patrones es muy baja.

Como puede verse en la tabla, cuando el corte relativo es 0.04 existen 45 patrones para los cuales nos se selecciona ningún patrón de entrenamiento. Dicho de otra forma, cuando se centra la hiperesfera de selección en alguno de estos 45 patrones, siendo su radio el 4% de la distancia máxima, ningún patrón de entrenamiento está situado en el interior de esta hiperesfera. También en este caso, se han excluido estos patrones de test del cómputo del error. En esta tabla puede verse que el error mínimo es 0.01554 y se obtiene con una red de 30 neuronas cuando se aplica un valor del corte relativo de 0.08.

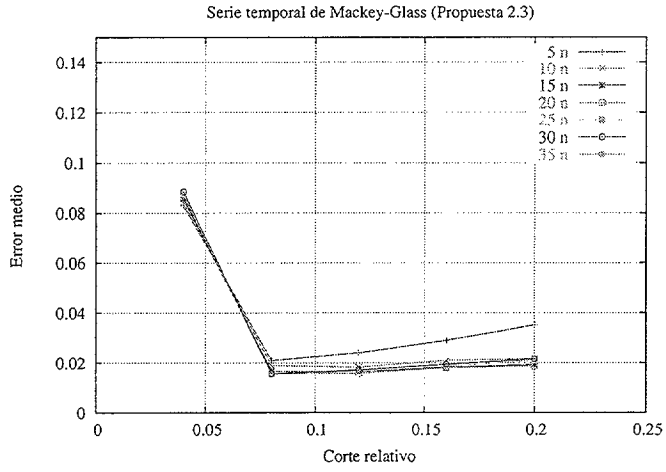


Figura 4.33:
Error medio con aprendizaje selectivo (Propuesta 2.3). Serie temporal de Mackey-Glass

En la figura 4.33 se representan gráficamente los resultados de la tabla 4.28, y puede verse que las curvas de error difieren bastante de las obtenidas en el dominio de las mareas de Venecia (ver figura 4.31), pues en la serie de Mackey-Glass cuando el corte es muy pequeño los errores son bastante grandes, aunque no se computen los patrones de test que producen "ceros". Esto es así porque con corte 0.04 sólamente hay 45 patrones, de un total de 500, para los que se da la situación anómala de ausencia de patrones de entrenamiento, y por tanto las curvas son más parecidas a las obtenidas en la propuesta 2.2 del método (ver figura 4.25).

Las características de los datos expuestos son similares a los del resto de los dominios y a las de la propuesta 2.2: cuando el corte aumenta y se selecciona un mayor número de patrones, el error medio disminuye, y si la red

tiene un número adecuado de neuronas, el error permanece prácticamente constante con el corte. En este caso, se ve que sólo cuando el número de neuronas es muy pequeño el error aumenta con el corte. Hay que destacar que, como en el resto de los dominios, se consiguen buenos resultados, que mejoran ligeramente a los obtenidos con el método 2.2, y son independientes de las inicializaciones de los centros.

Como en los dominios anteriores, también se ha realizado una comparación de los errores obtenidos con el método tradicional y con la propuesta 2.3, para cada uno de los 500 patrones del conjunto de test de este dominio. En ambos casos, se han elegido las arquitecturas que mejores resultados proporcionaban, así como el mejor corte relativo para el método selectivo.

En la figura 4.34 se muestran las representaciones gráficas de dichos errores. Puede observarse que para la gran mayoría de los 500 patrones de test, los errores de generalización producidos cuando se entrenan las redes por el método convencional son mayores que cuando se entrenan utilizando el método correspondiente a la propuesta 2.3.

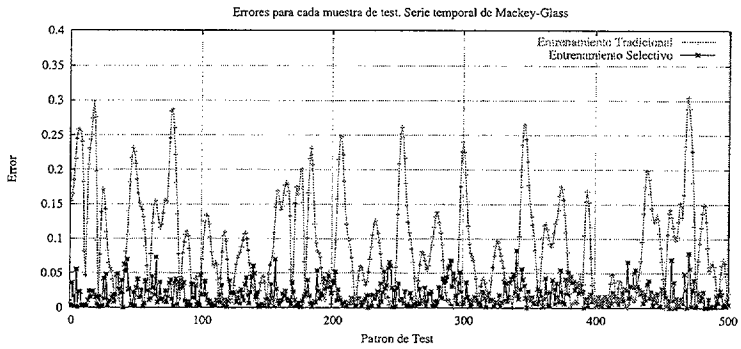


Figura 4.34:
Errores para cada patrón de test (Propuesta 2.3). Serie temporal de Mackey-Glass

Dominio de clasificación Pima-Indians Diabetes

Finalmente, se ha aplicado al dominio de clasificación Pima-Indians Diabetes el método correspondiente a la propuesta 2.3. Se han realizado los experimentos con el conjunto de test de este dominio con redes de 4, 6, ..., 16 neuronas y se han aplicado cortes relativos desde 0.12 hasta 0.44, con incrementos de 0.04. En la tabla 4.29 se muestran las tasas medias de aciertos obtenidas, es decir la proporción del número de patrones correctamente

clasificados sobre el total. Se han considerado como patrones correspondientes a la clase 0, aquellos para los que la salida de la red era inferior a 0.4, mientras que aquellos para los que la salida de la red era superior a 0.6 se han asignado a la clase 1.

Al igual que en los dominios de las Mareas de Venecia y Mackey-Glass, ocurre que para ciertos patrones de test no se selecciona ningún patrón de entrenamiento debido a la alta dimensionalidad del dominio y a la especial distribución de los patrones, cuando el corte es menor o igual que 0.32.

	Neuronas Ocultas							
Corte	4	6	8	10	12	14	16	Ceros
0.12	0.8214	0.8214	0.8095	0.8095	0.8214	0.8214	0.8333	108
0.16	0.7589	0.7872	0.7872	0.7376	0.7163	0.7305	0.7305	51
0.2	0.7651	0.759	0.747	0.7289	0.7169	0.7048	0.6988	26
0.24	0.7459	0.7238	0.7514	0.7072	0.7182	0.7017	0.7238	11
0.28	0.7128	0.6755	0.7181	0.7234	0.7394	0.7553	0.7713	4
0.32	0.7158	0.6895	0.7158	0.7158	0.6684	0.6789	0.6632	2
0.36	0.6615	0.7188	0.6979	0.7292	0.724	0.6979	0.6563	0
0.4	0.6615	0.6823	0.6927	0.6823	0.7135	0.6563	0.6719	0
0.44	0.6615	0.6979	0.7292	0.7031	0.6719	0.6719	0.75	0

Tabla 4.29:
Tasa de aciertos con aprendizaje selectivo. Propuesta 2.3. Pima-Indians Diabetes

Como en casos anteriores se incluye en la tabla la columna "Ceros" indicando el número de patrones de test para los cuales se da esta situación anómala. Se observa que el corte debe ser mucho más grande que en los casos anteriores para que con todas las muestras de test se seleccionen patrones de entrenamiento.

Se puede observar que con cortes muy pequeños se producen resultados aparentemente muy buenos, pero esto es así porque en la media no se han computado los patrones que producían "ceros". Éstos son los patrones más difíciles de generalizar, y es así porque se encuentran en zonas del espacio de entrada donde la densidad de patrones es muy baja, o dicho de otro modo, hay muy pocos patrones similares a ellos. Por tanto, los datos obtenidos para los cortes que producen "ceros" no son significativos y no deben ser tenidos en cuenta. Del resto de los datos destaca la tasa de aciertos de 0.75 producida con una red de 16 neuronas y un corte de 0.44.

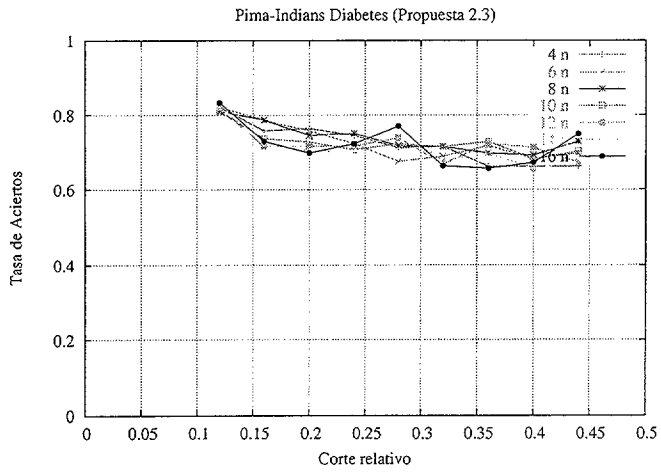


Figura 4.35:
Tasa de aciertos con aprendizaje selectivo. Propuesta 2.3. Pima-Indians Diabetes

En la figura 4.35 se representan gráficamente los datos de la tabla.

Comparación de los resultados con las fases anteriores

Una vez vistos los resultados de la experimentación con cada uno de los dominios, en la tabla 4.30 se muestran los mejores resultados obtenidos con la propuesta 2.3 junto con los obtenidos con las propuestas anteriores y con el método tradicional, para los dominios estudiados. En esta tabla puede verse que los mejores resultados obtenidos en esta propuesta son equivalentes a los obtenidos en la propuesta 2.2, aunque con la ventaja de que no es necesario realizar diferentes inicializaciones aleatorias de los centros, y de esta forma los resultados no dependen de dicha inicialización.

Conclusiones

Una fase muy importante del entrenamiento de las RNBR es la que fija los centros de las neuronas, y esto normalmente se hace utilizando el algoritmo K-medias. Este algoritmo alcanza mínimos locales dependiendo de la inicialización de los centros. La versión clásica de este algoritmo inicializa los centros aleatoriamente, y por tanto las RNBR alcanzan resultados bastante diferentes en distintas simulaciones, dependiendo de las inicializaciones aleatorias del algoritmo K-medias, como se comentó en las conclusiones de

Error Medio o Tasa de aciertos	Propuesta 2.3	Propuesta 2.2	Propuesta 2.1	Propuesta 1	Entrenam. tradicional
Función definida por partes	0.00208 $r_r = 0.06$ 11 n	0.00273 $r_r = 0.06$ 19 n	0.016087 4 neuronas	0.0233 σ=0.25 9 n	0.04156 100 n
Polinomio de Hermite	0.0038 $r_r = 0.12$ 11 n	0.00345 $r_r = 0.08$ 9 n	0.00571 10 n	0.0121 σ=0.1 5 n	0.01904 40 n
Serie de las Mareas de Venecia	0.02059 $r_r = 0.16$ 19 n	0.02888 $r_r = 0.12$ 23 n	0.04117 20 n	0.0905 σ=0.15 17 n	0.0605 50 n
Serie de Mackey-Glass	0.01554 $r_r = 0.08$ 30 n	0.01651 $r_r = 0.15$ 25 n	0.02429 32 n	0.0348 σ=0.1 9 n	0.10273 110 n
Pima-Indians Diabetes	0.75 $r_r = 0.44$ 16 n				0.73274 50 n

Tabla 4.30:
Comparación de los mejores resultados obtenidos con entrenamiento selectivo (Propuesta 1, 2.1, 2.2 y 2.3) y con entrenamiento tradicional

la propuesta 2.2. En la propuesta 2.3 se modifica el método de aprendizaje selectivo, para evitar que los resultados dependan de las inicializaciones aleatorias del algoritmo K-medias, realizando una inicialización determinista de los centros. Las conclusiones obtenidas al analizar los datos obtenidos en la experimentación son las siguientes:

- Con el método propuesto se obtienen buenos resultados, equivalentes a los obtenidos en la propuesta 2.2. Lo importante es que ya no es necesario hacer varias simulaciones para evitar caer en mínimos locales especialmente malos debido a las inicializaciones aleatorias de K-medias. Al realizar una inicialización determinista, siempre se obtienen los mismos resultados.

- Al realizar los experimentos con los dominios de las Mareas de Venecia, la serie de Mackey-Glass y Pima-Diabetes, caracterizados por su alta dimensionalidad, se ha observado que cuando el corte relativo utilizado es pequeño, no se seleccionan patrones de entrenamiento para algunos patrones de test. Esto es debido a que cuando el número de dimensiones es elevado, las distancias entre patrones son más grandes. Además, por las características de los dominios estudiados, la distribución de los patrones en el espacio de entrada no es uniforme sino

que hay zonas donde la densidad de patrones es muy baja. Esto es especialmente llamativo en el dominio de clasificación Pima-Indians Diabetes, donde existen patrones de test de estas características incluso cuando el corte alcanza el valor de 0.32. Dicho de otra forma, si centramos una hiperesfera con un radio del 32 % de la distancia máxima en algunos patrones de test, no se encontraría ningún patrón de entrenamiento en su interior.

4.4.4 Tratamiento especial de patrones de test para los que no se seleccionan patrones de entrenamiento

Una de las conclusiones más importantes de la propuesta 2.3 es que, en ciertos dominios, existen patrones de test que están situados en zonas de muy baja densidad de patrones, o lo que es equivalente, existen muy pocos patrones de entrenamiento similares a esos patrones de test. Cuando la dimensión de los datos de entrada es alta, las distancias entre patrones son más grandes, por lo que es muy habitual que con cortes relativos pequeños no se seleccione ningún patrón de entrenamiento. Dicho de otro modo, si el radio de la hiperesfera de selección es pequeño, al situar su centro en patrones de test pertenecientes a zonas de baja densidad de patrones, no se encontrará ningún patrón de entrenamiento en su interior. En los resultados expuestos en la propuesta anterior, dichos patrones de test no eran evaluados.

En esta propuesta, denominada *propuesta 2.4*, se proponen dos métodos alternativos que modifican la propuesta 2.3 para que la red pueda responder a estos patrones de test especiales.

- **Método 1.** Si para un patrón de test $\mathbf{q}$, el conjunto de patrones seleccionados X_q está vacío, entonces se aplica el método de selección de patrones al patrón de entrenamiento más cercano, como si fuera el patrón de test. Es decir:

  1. Sea $\mathbf{x_c}$ el patrón de entrenamiento más cercano a $\mathbf{q}$. Entonces se considerará $\mathbf{x_c}$ como el nuevo patrón de test: $\mathbf{q}' \leftarrow \mathbf{x_c}$
  2. Se aplica al patrón $\mathbf{q}'$ el método de aprendizaje selectivo con ponderación inversa, según la propuesta 2.3. Es decir, se sitúa el centro de la hiperesfera de selección en el patrón $\mathbf{q}'$ y se genera el conjunto $X_{q'}$. Como $\mathbf{q}' \in X$, siendo X el conjunto de entrenamiento original, siempre habrá al menos un patrón en $X_{q'}$.
  3. Se entrena la red con el conjunto X_q para responder al patrón $\mathbf{q}$ tal como se hacía en las propuestas anteriores.

- **Método 2.** Si para un patrón de test $\mathbf{q}$, el conjunto de patrones seleccionados X_q está vacío, entonces se entrena la red utilizando todos los patrones de entrenamiento del conjunto original X. Es decir, para este patrón de test $X_q = X$.

Resultados Experimentales

Se ha aplicado la propuesta 2.4 con sus dos variantes a los dominios donde se producía la situación anómala de ausencia de patrones de entrenamiento

para ciertos patrones de test con determinados cortes relativos: serie temporal de las Mareas de Venecia, serie temporal de Mackey-glass y dominio de clasificación Pima-Indians Diabetes. Esta propuesta sólo difiere de la propuesta 2.3 cuando se desea generalizar estos patrones de test especiales y por tanto, los valores de error medio o tasa de aciertos obtenidos cuando no existen estos patrones de test, son idénticos a los que se habían obtenido en la propuesta 2.3. A continuación se describen los experimentos realizados con los tres dominios citados.

Mareas de Venecia. Método 1

En la tabla 4.31 se muestran los resultados obtenidos al aplicar el método 1, con redes de $3, 5, \ldots, 15$ neuronas y cortes relativos de $0.04, 0.08, \ldots, 0.2$. Este método sólo tiene efecto cuando para un patrón de test no se seleccionan patrones de entrenamiento, por lo que los resultados obtenidos para cortes mayores o iguales que 0.12 son exactamente iguales que los obtenidos en la propuesta 2.3, como puede verse en la tabla 4.27. Ahora ya no se producen "ceros" por lo que la columna correspondiente se ha eliminado. El mejor valor obtenido en este experimento sigue siendo el error 0.02059 obtenido en la red de 19 neuronas cuando se aplica un corte relativo de 0.16.

	Neuronas Ocultas						
Corte	3	7	11	15	19	23	27
0.04	0.06004	0.06042	0.06276	0.06292	0.06186	0.06330	0.06352
0.08	0.04299	0.03685	0.03447	0.03011	0.03197	0.02792	0.03231
0.12	0.05457	0.02967	0.02682	0.02269	0.02234	0.02235	0.02643
0.16	0.07463	0.02869	0.02398	0.02913	0.02059	0.02514	0.02552
0.2	0.08459	0.03769	0.02420	0.02411	0.02728	0.02288	0.03336

Tabla 4.31:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 1). Serie temporal de las Mareas de Venecia

En la figura 4.36, se representan gráficamente los resultados de la tabla 4.31. En ella se puede ver el moderado error que se produce con cortes 0.04 y 0.08. Hay que tener en cuenta que en la propuesta 2.3 los datos producidos para estos cortes no eran significativos, porque no se tenían en cuenta todos los patrones de test. Sin embargo, para el resto de los cortes las curvas son idénticas a las de la propuesta 2.3.

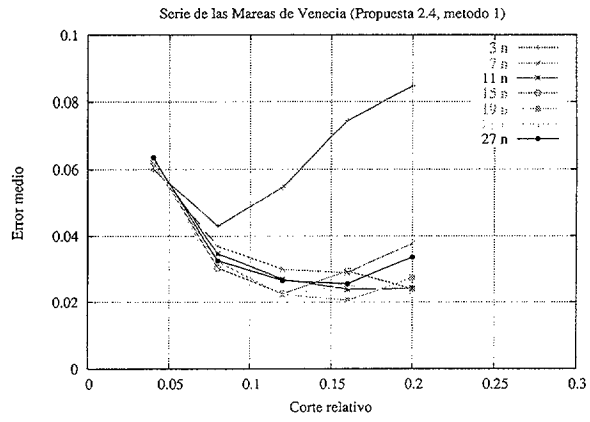


Figura 4.36:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 1). Serie temporal de las Mareas de Venecia

Mareas de Venecia. Método 2

Se ha aplicado al mismo dominio el método 2 de la propuesta 2.4, donde se entrena la red con todo el conjunto de entrenamiento cuando para un patrón de test el conjunto seleccionado está vacío. El número de neuronas y los cortes elegidos son los mismos que en el experimento anterior. En la tabla 4.32 se muestran los datos obtenidos, siendo el mejor valor el error 0.02059, igual que en el método anterior y en la propuesta 2.3, producido por una red de 19 neuronas y un corte relativo de 0.16.

	Neuronas Ocultas						
Corte	3	7	11	15	19	23	27
0.04	0.10215	0.09542	0.08128	0.06672	0.06239	0.06333	0.06500
0.08	0.04695	0.04497	0.04382	0.03572	0.03407	0.03266	0.03441
0.12	0.05457	0.02967	0.02682	0.02269	0.02234	0.02235	0.02643
0.16	0.07463	0.02869	0.02398	0.02913	0.02059	0.02514	0.02552
0.2	0.08459	0.03769	0.02420	0.02411	0.02728	0.02288	0.03336

Tabla 4.32:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 2). Serie temporal de las Mareas de Venecia

En la figura 4.37 puede verse la representación gráfica de los datos de la tabla 4.32. Las curvas sólo se diferencian en los cortes 0.04 y 0.08, y cabe

destacar que cuando se utiliza el método 2 los errores obtenidos varían más con las arquitecturas, especialmente con el corte 0.04.

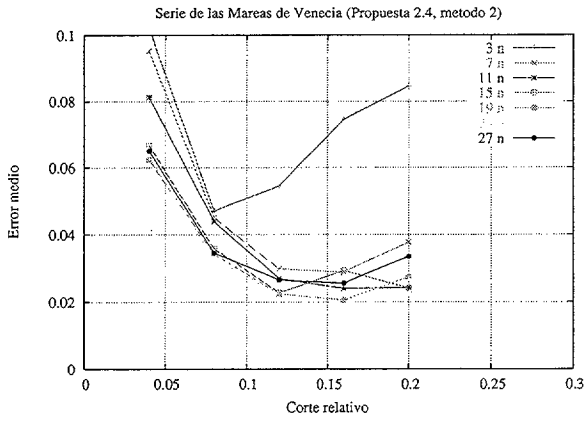


Figura 4.37:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 2). Serie temporal de las Mareas de Venecia

En la tabla 4.33 se comparan los resultados obtenidos para los cortes 0.04 y 0.08 con los del experimento anterior, y se observa que, en general, los errores obtenidos con el método 2 de la propuesta 2.4 son mayores que los obtenidos por el método 1 de dicha propuesta. Es decir, en el dominio de las mareas de Venecia cuando los cortes son muy pequeños y se producen "ceros" es mejor entrenar las redes con los patrones cercanos al patrón más cercano al de test, aunque este patrón más cercano no sea muy similar a él (método 1), que entrenar la red con todos los patrones de entrenamiento (método 2).

		Neuronas Ocultas						
Corte	Prop	3	7	11	15	19	23	27
0.04	2.4(2)	0.10215	0.09542	0.08128	0.06672	0.06239	0.06333	0.06500
	2.4(1)	0.06004	0.06042	0.06276	0.06292	0.06186	0.06330	0.06352
0.08	2.4(2)	0.04695	0.04497	0.04382	0.03572	0.03407	0.03266	0.03441
	2.4(1)	0.04299	0.03685	0.03447	0.03011	0.03197	0.02792	0.03231

Tabla 4.33:
Comparación de errores utilizando los dos métodos de la propuesta 2.4. Serie temporal de las Mareas de Venecia

Mackey-Glass. Método 1

También se ha aplicado el método 1 de la propuesta 2.4 al dominio de Mackey-Glass. En este dominio, cuando se utiliza un corte de 0.04, existen 45 patrones de test, de un total de 500, para los que el conjunto de entrenamiento asociado está vacío. Se han utilizado redes de 5, 10, ..., 30 neuronas y cortes relativos de 0.04, 0.08, ..., 0.2. Obviamente, los errores obtenidos para cortes mayores de 0.04 serán idénticos a los obtenidos en la propuesta 2.3.

	Neuronas Ocultas					
Corte	5	10	15	20	25	30
0.04	0.02866	0.03081	0.03132	0.03205	0.03380	0.03349
0.08	0.02092	0.01897	0.01666	0.01564	0.01561	0.01554
0.12	0.02415	0.01821	0.01644	0.01571	0.01650	0.01711
0.16	0.02917	0.02099	0.01812	0.01802	0.01847	0.01936
0.2	0.03522	0.02169	0.01927	0.01866	0.02109	0.02156

Tabla 4.34:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 1). Serie temporal de Mackey-Glass

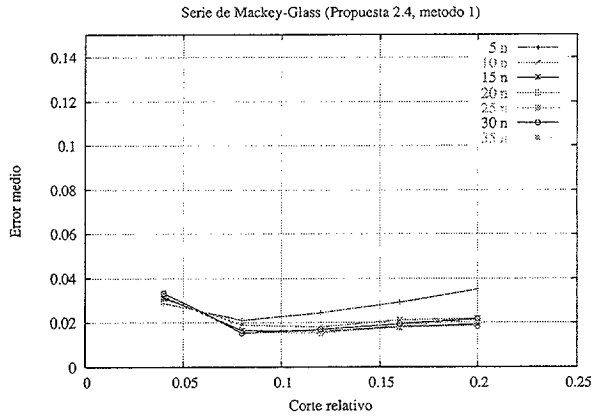


Figura 4.38:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 1). Serie temporal de Mackey-Glass

La tabla 4.34 contiene los errores medios obtenidos para cada corte y cada arquitectura, siendo el mejor valor el error 0.01554, producido por una

red de 30 neuronas cuando se utiliza un corte relativo de 0.08, y por tanto es igual al error producido para esa arquitectura y corte cuando se aplica la propuesta 2.3.

En la figura 4.38 se representan gráficamente los datos de la tabla 4.34. Las curvas de error son idénticas a las de la gráfica 4.33 de la propuesta 2.3 excepto en el corte 0.04. Se ve que el comportamiento de las distintas arquitecturas con este corte es muy similar.

Mackey-Glass. Método 2

También se aplica el método 2 de la propuesta actual al dominio de Mackey-Glass, utilizando las mismas arquitecturas y cortes relativos que en el experimento anterior. En el método 2, cada vez que un patrón de test produce un conjunto de entrenamiento asociado vacío, se entrena la red con el conjunto total de patrones de entrenamiento. Los resultados obtenidos se muestran en la tabla 4.35, donde el mejor valor sigue siendo el error 0.01554, para la red de 30 neuronas y corte de 0.08, idéntico al producido en el método 1 de esta propuesta y en la propuesta 2.3.

	Neuronas Ocultas					
Corte	5	10	15	20	25	30
0.04	0.03277	0.03527	0.03545	0.03405	0.03372	0.03605
0.08	0.02092	0.01897	0.01666	0.01564	0.01561	0.01554
0.12	0.02415	0.01821	0.01644	0.01571	0.01650	0.01711
0.16	0.02917	0.02099	0.01812	0.01802	0.01847	0.01936
0.2	0.03522	0.02169	0.01927	0.01866	0.02109	0.02156

Tabla 4.35:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 2). Serie temporal de Mackey-Glass

En la figura 4.39 se muestra la representación gráfica de los datos completos de la tabla 4.35. Las curvas del error son muy similares a las de la gráfica 4.34, correspondientes al método 1, ya que en el único corte donde se producen diferencias éstas son muy pequeñas como ya se ha comentado. En ambos casos, el tratamiento de los patrones de test anómalos produce unos resultados satisfactorios.

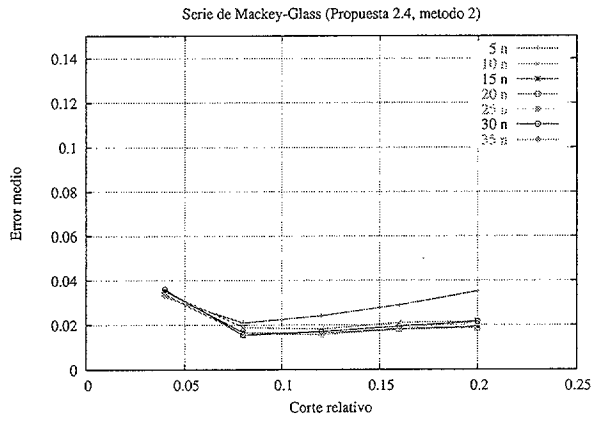


Figura 4.39:
Errores medios con aprendizaje selectivo (Propuesta 2.4, método 2). Serie temporal de Mackey-Glass

Pima-Indians Diabetes. Método 1

Como se vio en el apartado 4.4.3, al aplicar el método correspondiente a la propuesta 2.3 con cortes menores que 0.32 existían patrones de test cuyos conjuntos de entrenamiento asociados estaban vacíos. Este dominio tiene características especiales si se compara con el resto, pues esta situación anómala se produce con cortes relativamente grandes. En la tabla 4.36 puede verse, para cada corte relativo, el número de patrones de test para los cuales no se selecciona ningún patrón de entrenamiento.

Corte	Número de Ceros
0.12	108
0.16	51
0.2	26
0.24	11
0.28	4
0.32	2
0.36	0

Tabla 4.36:
Número de patrones de test anómalos. Propuesta 2.4. Pima-Indians Diabetes

Es llamativo el hecho de que para un corte tan grande como 0.32 existan

dos "ceros", o dicho de otro modo, que existen en el conjunto de test dos patrones tales que si centramos en ellos una hiperesfera cuyo radio es el 32 % de la distancia al patrón de entrenamiento más alejado, no existiría ningún patrón de entrenamiento en su interior. Por supuesto, a medida que el corte disminuye el número de ceros aumenta, de forma que con un corte de 0.12 el número de patrones especiales es 108, de un total de 192. Este comportamiento del dominio se explica por su alta dimensionalidad (8 dimensiones) y por la existencia de zonas del espacio de entrada donde la densidad de patrones es muy baja.

Se ha aplicado el método 1 de esta propuesta al conjunto de datos representativo del dominio, con redes de 4, 6, ..., 16 neuronas, y cortes relativos que varían desde 0.12 hasta 0.44, con incrementos de 0.04. En la tabla 4.37 se muestran los valores de las tasas de aciertos medias obtenidas. El corte 0.32 era el máximo con el que se producía la situación anómala de patrones de test para los que no se seleccionaban patrones de entrenamiento, por tanto los valores correspondientes a cortes mayores de 0.32 son idénticos a los obtenidos en la propuesta 2.3. Ahora, todos los valores obtenidos son significativos, pues las redes han respondido a todas las muestras de test. Las tasas de aciertos obtenidas cuando los cortes son pequeños no son altas, y esto indica que el método 1 de la propuesta 2.3 no es muy apropiado para este dominio.

	Neuronas Ocultas						
Corte	4	6	8	10	12	14	16
0.12	0.6354	0.6354	0.6458	0.6458	0.6406	0.6354	0.6406
0.16	0.6667	0.6354	0.6354	0.625	0.6458	0.6302	0.625
0.2	0.7135	0.7083	0.7083	0.6927	0.7031	0.6719	0.6719
0.24	0.7396	0.724	0.7188	0.6823	0.6771	0.7083	0.6823
0.28	0.6875	0.6875	0.7083	0.651	0.7188	0.6979	0.743
0.32	0.7292	0.6615	0.7188	0.7135	0.651	0.7031	0.6458
0.36	0.6615	0.7188	0.6979	0.7292	0.724	0.6979	0.6563
0.4	0.6615	0.6823	0.6927	0.6823	0.7135	0.6563	0.6719
0.44	0.6615	0.6979	0.7292	0.7031	0.6719	0.6719	0.75

Tabla 4.37:
Tasa de aciertos con aprendizaje selectivo (Propuesta 2.4, método 1).
Pima-Indians Diabetes

En la figura 4.40 se muestra la representación gráfica de dichos datos, observándose la tendencia de una cierta mejora en las tasas de aciertos cuando aumenta el corte. Puede verse tanto en la tabla como en la gráfica que el mejor valor sigue siendo el de 0.75 para una red de 16 neuronas cuando

se utiliza un corte de 0.44.

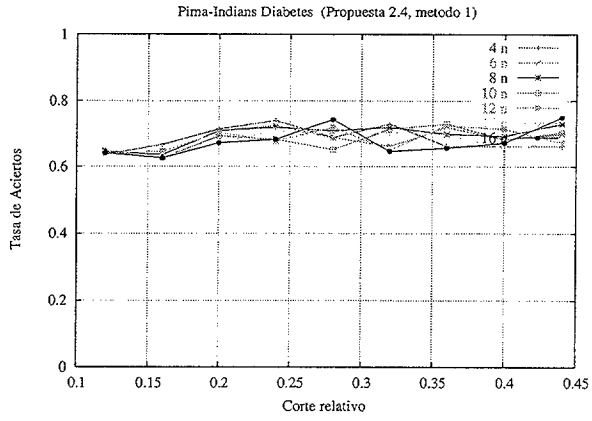


Figura 4.40:
Tasa de aciertos con aprendizaje selectivo (Propuesta 2.4, método 1).
Pima-Indians Diabetes

Pima-Indians Diabetes. Método 2

Igual que en los dominios anteriores, se ha aplicado el método 2 al mismo conjunto de datos, con redes desde 4 a 16 neuronas, con incrementos de 2 neuronas, y cortes desde 0.12 hasta 0.44, con incrementos de 0.04. En la tabla 4.38 se muestran los resultados obtenidos y en la figura 4.41 su representación gráfica.

	Neuronas Ocultas						
Corte	4	6	8	10	12	14	16
0.12	0.6563	0.6927	0.6927	0.7083	0.7344	0.7292	0.7344
0.16	0.7031	0.7604	0.7656	0.724	0.6927	0.7135	0.7135
0.2	0.7344	0.7396	0.7396	0.6979	0.7188	0.6979	0.6823
0.24	0.7135	0.7031	0.7135	0.6979	0.7031	0.6823	0.7083
0.28	0.6927	0.651	0.7031	0.7135	0.7344	0.7448	0.7656
0.32	0.7135	0.6927	0.7083	0.724	0.6563	0.6771	0.6406
0.36	0.6615	0.7188	0.6979	0.7292	0.724	0.6979	0.6563
0.4	0.6615	0.6823	0.6927	0.6823	0.7135	0.6563	0.6719
0.44	0.6615	0.6979	0.7292	0.7031	0.6719	0.6719	0.75

Tabla 4.38:
Tasa de aciertos con aprendizaje selectivo (Propuesta 2.4, método 2).
Pima-Indians Diabetes

Puede observarse que los resultados obtenidos para cortes pequeños mejoran considerablemente respecto al método 1. De hecho, ahora el mejor valor obtenido es una tasa de aciertos de 0.7656, que corresponde a una red de 8 neuronas para un corte relativo de 0.16, con el que se producían 51 ceros. Este comportamiento es diferente al de los dominios anteriores, donde el método 1 se comportaba mejor que el método 2. Se puede explicar por las especiales características de este dominio, donde patrones cercanos en el espacio de entrada pueden pertenecer a clases diferentes. Cuando se tiene un patrón de test "anómalo", si se entrena la red con el conjunto completo su rendimiento en general será mejor que si se entrena sólamente con los patrones más cercanos al patrón más cercano al de test, que muy bien pueden pertenecer a la clase opuesta.

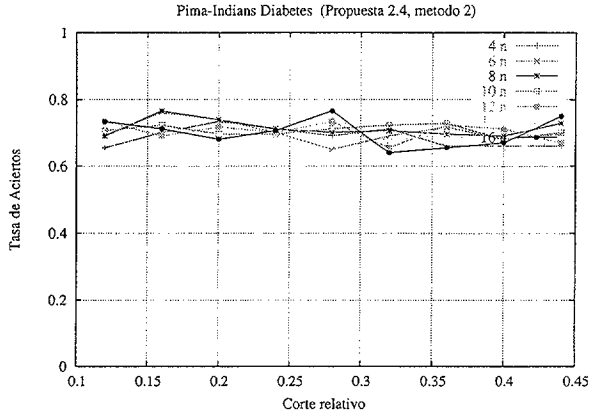


Figura 4.41:
Tasa de aciertos con aprendizaje selectivo (Propuesta 2.4, método 2).
Pima-Indians Diabetes

Conclusiones

En la propuesta 2.4 se presentan dos métodos para solucionar el problema detectado en la propuesta 2.3 que aparece en dominios de alta dimensionalidad y con una especial distribución de sus datos. En ellos puede ocurrir que para ciertos patrones de test situados en zonas de baja densidad de datos, al utilizar cortes pequeños no se seleccione ningún patrón de entrenamiento. El primer método utiliza los patrones de entrenamiento incluidos en la hiperesfera centrada en el patrón de entrenamiento más próximo al patrón de test anómalo mientras que el segundo método utiliza todos los patrones de entrenamiento.

Se han realizado los experimentos aplicando los dos métodos a los dominios afectados: las series temporales de las mareas de Venecia y Mackey-Glass y el dominio de clasificación Pima-Indians Diabetes, y los resultados muestran lo siguiente:

- Ambos métodos solucionan un problema no tratado en las propuestas anteriores. Cuando para cortes pequeños existen patrones de test anómalos, los errores obtenidos con los dos métodos de la propuesta 2.4 son totalmente válidos, mientras que los obtenidos en las propuestas anteriores no podían ser tenidos en cuenta.

- En las mareas de Venecia y en la serie de Mackey-Glass, el método 1 se comporta mejor que el método 2 cuando existen patrones anómalos. Por tanto, es mejor entrenar las redes con los patrones más cercanos al patrón de entrenamiento más cercano al de test, que entrenarlas con todos los patrones. Sin embargo, en el dominio Pima-Indians Diabetes ocurre lo contrario. Esto es debido a las especiales características del dominio, en el cual existen bastantes patrones cercanos en el espacio de entrada que pertenecen a clases diferentes. Cuando para un patrón de test no se seleccionan patrones de entrenamiento, es porque está en una zona de muy baja densidad de patrones, y si se entrena la red con el conjunto completo, su rendimiento, en general, será mejor que si se entrena solamente con los patrones más cercanos al patrón más cercano al de test, puesto que pueden pertenecer a la clase opuesta.

Capítulo 5

Conclusiones y líneas futuras de investigación

En esta tesis se ha presentado un modelo de aprendizaje retardado aplicado al entrenamiento de RNBR. Como se ha visto en el capítulo 2, las RNBR son aproximadores universales, en el sentido de que pueden aproximar cualquier relación no lineal entre la entrada y la salida. Son modelos robustos, tolerantes al ruido y a los atributos irrelevantes. Además, su entrenamiento es muy rápido en comparación con otros modelos de redes de neuronas, debido al carácter local de estas redes y al carácter lineal de su capa de salida. Su principal inconveniente reside en la pobre capacidad de generalización, ya que requieren un número grande de unidades ocultas, especialmente si la dimensionalidad del espacio de entrada es alta, lo cual influye negativamente en su capacidad de generalización. Por otra parte, los métodos de aprendizaje retardado pueden tener una buena capacidad de generalización pues construyen las representaciones de la función objetivo de forma local dependiendo de la nueva muestra de test, pero la precisión de la generalización depende del número de patrones que se seleccionen. Además, en estos métodos la calidad de la generalización también depende mucho de qué instancias son consideradas más cercanas a la muestra a generalizar, lo cual está determinado por la función de distancia utilizada. Puede darse el caso de instancias donde los atributos relevantes son muy parecidos y sin embargo están muy distantes entre sí porque difieren bastante en atributos poco importantes. Cuando ocurre esto, los métodos de aprendizaje retardado no consiguen generalizar adecuadamente.

En esta tesis, se propone un método basado en técnicas de aprendizaje retardado para entrenar RNBR, de forma que cuando se recibe una mues-

tra de test, se seleccionan los patrones más relevantes -más similares a esa muestra de test- con los cuales se formará un conjunto de entrenamiento con el que la red deberá entrenarse y así responder adecuadamente a ese patrón de test. Sobre el método básico, se han desarrollado dos propuestas denominadas *Ponderación Gaussiana* y *Ponderación Inversa*. La primera utiliza una función gaussiana como función kernel para ponderar los patrones de entrenamiento en función de su distancia a la muestra de test, mientras que la segunda utiliza una función inversa para realizar dicha ponderación. A la segunda propuesta se le han incorporado varias modificaciones dando lugar a cuatro variantes del método.

Para validar el método propuesto, se han realizado experimentos sobre cinco dominios para medir la capacidad de generalización del modelo frente a la de RNBR entrenadas por el método convencional.

5.1 Conclusiones

Las principales conclusiones que extraen cuando se analiza globalmente el método propuesto se exponen a continuación.

Cuando se entrenan las RNBR utilizando una selección de patrones de entrenamiento, su capacidad de generalización aumenta. La selección de los patrones más relevantes permite obtener RNBR capaces de aproximar funciones complejas con mayor precisión.

De las dos funciones kernel utilizadas para seleccionar los patrones de entrenamiento, la función inversa es la que produce los mejores resultados. Además, la función gaussiana tiene el inconveniente del parámetro σ, habiéndose comprobado que el comportamiento de las redes depende en gran medida de este parámetro. Este inconveniente no existe con la ponderación inversa puesto que la función no depende de ningún parámetro. Aunque en las propuestas correspondientes a la ponderación inversa se ha introducido el parámetro corte, que indica la extensión de la región de vecindad donde se realizará la selección de los patrones de entrenamiento, se ha comprobado que la capacidad de generalización de las RNBR apenas se ve influenciada por este parámetro, siempre que la región de vecindad supere un tamaño mínimo y que la red posea un número suficiente de neuronas.

Un importante inconveniente de los métodos de aprendizaje retardado es el alto coste computacional. En el método propuesto el coste computacional es grande debido a que cada vez que se presenta una muestra de test hay que entrenar la red, pero esto no es una desventaja del método puesto que se consiguen precisiones en la generalización que el método convencional no

es capaz de lograr. En muchas aplicaciones lo importante es que el error en la generalización sea lo más bajo posible, sin ser el tiempo un factor crucial. De todos modos, el número de neuronas necesarias para realizar el entrenamiento selectivo es mucho menor que el necesario cuando se utilizan todos los patrones de entrenamiento, con lo que el esfuerzo computacional no es tan alto como parece a primera vista.

Otra importante conclusión es que el método propuesto es aplicable a otros modelos de redes de neuronas, pues no está ligado a un modelo concreto aunque se haya validado solamente con RNBR.

De forma más pormenorizada se expondrán otras conclusiones obtenidas del análisis de los resultados en cada una de las propuestas del método.

Con la ponderación gaussiana se mejoran ligeramente los resultados obtenidos por las RNBR entrenadas de forma convencional, debido a que la red se entrena solamente con los patrones más similares al nuevo patrón de test. Aunque el coste computacional es mayor porque hay que entrenar la red para cada patrón de test, el número de neuronas necesario es sensiblemente menor. El error de generalización obtenido depende mucho de la desviación, obteniéndose resultados aceptables dentro de un margen relativamente estrecho de la desviación. Cuando ésta es pequeña, el error es grande debido a que se seleccionan muy pocos patrones de entrenamiento, aunque se repitan muchas veces. Cuando ésta crece por encima de cierto valor, dejan de seleccionarse la mayor parte de los patrones de entrenamiento, por lo que el error aumenta de forma muy acusada.

Cuando se utiliza la ponderación inversa, los resultados obtenidos mejoran los obtenidos por el método tradicional con un amplio rango de arquitecturas, haciendo que el método sea robusto pues tiene cierto grado de independencia de la arquitectura. En la ponderación gaussiana la dependencia de la arquitectura era mayor, y además existía una fuerte dependencia del parámetro desviación.

El valor de redondeo, utilizado en la primera propuesta de la ponderación inversa (propuesta 2.1) para transformar en frecuencias enteras los valores de la función kernel, apenas tiene influencia en los resultados. Además, al inicializar los centros según la versión clásica del algoritmo K-medias se observa que, en ocasiones y dependiendo de las inicializaciones aleatorias, hay clases del espacio de entrada que permanecen vacías. Esto hace que el comportamiento de la red empeore.

En la segunda propuesta del método de ponderación inversa (propuesta 2.2) , donde se introduce el parámetro *corte*, los resultados obtenidos son significativamente mejores que los obtenidos en las propuestas anteriores y cuando se entrenan las redes por el método convencional. Además, se

manifiesta la robustez del método, pues es relativamente independiente de la arquitectura de las redes y del corte elegido, siempre que estos parámetros estén dentro de un rango relativamente amplio. El parámetro *corte* utilizado no es un parámetro crítico, ya que a partir de un valor lo suficientemente grande para que se seleccione un número mínimo de patrones, el error apenas varía, siempre que la red tenga un número suficiente de neuronas. También en esta propuesta se pone de manifiesto que el comportamiento de las RNBR dependen significativamente de las inicializaciones aleatorias del algoritmo K-medias.

En la tercera propuesta del método de ponderación inversa, donde se modifica el método realizando una inicialización determinista de los centros en el algoritmo K-medias, los resultados obtenidos muestran que la capacidad de generalización conseguida es similar a la de la propuesta anterior, pero ya no es necesario realizar varias simulaciones, puesto que el método es determinista y siempre se obtienen los mismos resultados. En dominios de alta dimensionalidad y con una especial distribución de los patrones en el espacio de entrada, puede ocurrir que para ciertos patrones de test no se seleccione ningún patrón de entrenamiento cuando la región de vecindad es pequeña, con lo que la red produciría resultados arbitrarios para esos patrones de test anómalos.

Para solucionar este problema se introduce la cuarta propuesta del método de ponderación inversa donde se indican dos algoritmos alternativos denominados método 1 y método 2. En los dominios de series temporales, cuando se da la situación descrita en el párrafo anterior, el método 1, que efectúa una selección de patrones alrededor del vecino más próximo al patrón de test, ofrece mejores resultados que el método 2, donde se entrena la red con todos los patrones disponibles. Es decir, también en estos casos especiales es mejor entrenar la red con una selección de patrones relevantes que con todos los patrones disponibles. Sin embargo, en el dominio de clasificación utilizado el método 2 proporciona mejores resultados. Esto se explica por las especiales características del dominio donde ocurre con frecuencia que patrones cercanos en el espacio de entrada pertenecen a la clase opuesta.

5.2 Líneas futuras de Investigación

El trabajo desarrollado en esta tesis plantea líneas que pueden ser estudiadas, desarrolladas y verificadas en el futuro. Entre ellas se proponen las siguientes:

Estudio del comportamiento de diferentes funciones kernel, para realizar

la selección de los patrones de entrenamiento. Entre estas funciones posibles se proponen diferentes funciones hiperbólicas, exponenciales y lineales. Todas ellas deben tener las características que deben poseer las funciones kernel: su valor será máximo cuando la distancia sea cero y a medida que la distancia aumenta el valor de la función debe decrecer sin discontinuidades. Estas funciones podrán poseer parámetros cuya influencia en la capacidad de generalización de las redes deberá estudiarse.

También puede plantearse la búsqueda automática de la función kernel óptima para un cierto problema. Esta búsqueda puede hacerse utilizando programación genética, estableciendo una serie de funciones primitivas de forma que se encuentre la función que minimice el error medio sobre un conjunto de validación extraído del conjunto de entrenamiento. A la vista de los resultados de la presente tesis, parece conveniente aplicar el parámetro corte para delimitar la región de vecindad. Este corte podría fijarse a un valor suficiente para que se seleccionaran suficientes patrones de entrenamiento. Una vez determinada la función óptima habría que validarla sobre el verdadero conjunto de test. El inconveniente que tendría este método sería el elevado coste computacional, aunque una vez determinada la función óptima, el coste sería equivalente al del método propuesto en esta tesis.

Otra línea que debe ser tenida en cuenta es el estudio de nuevas funciones de distancia, diferentes de la distancia euclídea. En una primera etapa debería estudiarse la distancia euclídea diagonalmente ponderada, donde interviene una matriz diagonal donde los coeficientes son los factores de escala de cada una de las dimensiones del espacio de entrada. Estos coeficientes de la matriz podrían determinarse utilizando algoritmos genéticos. El procedimiento sería el siguiente: se extraería un conjunto de validación del conjunto de entrenamiento y se utilizaría el algoritmo genético para determinar los coeficientes de la matriz que minimizaran el error medio sobre el conjunto de validación. Habría que aplicar el método de ponderación inversa, pues es el que mejores resultados ha ofrecido en esta tesis, y habría que fijar el corte a un valor lo suficientemente alto para que no influyera en los resultados. Una vez determinados los pesos, habría que utilizar dicha función de distancia para validar el método sobre el verdadero conjunto de test. El siguiente paso natural sería generalizar la distancia anterior, utilizando la distancia de Mahalanobis, donde la matriz ya no es diagonal sino que puede ser arbitraria. Ahora el número de coeficientes a determinar no es n sino n*n, y el coste computacional sería más elevado.

Una continuación natural de esta tesis es la validación del método de vecindad ponderada en otros modelos de redes de neuronas, especialmente en el perceptron multicapa. En este tipo de red el coste computacional sería

bastante más elevado, pero hay problemas donde el tiempo no es crítico y lo verdaderamente importante es conseguir mejores resultados en la generalización. La decisión de cuándo parar el entrenamiento en el perceptron multicapa es un problema importante y existen varias técnicas para resolverlo. Al entrenar el perceptron utilizando el método selectivo este problema es mayor puesto que hay que decidir cuándo parar para cada patrón de test, y no una sola vez para todo el conjunto como ocurre cuando se entrena el perceptron por el método tradicional.

Bibliografía

[Aha, 1973] Aha, D. (1973). Efficient instance-based algorithms for learning numeric functions. *Unpublished N.Y: Wiley, 1973.*

[Aha et al., 1991] Aha, D., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Bennett and Blue, 1997] Bennett, K. and Blue, J. (1997). A support vector machine approach to decision trees. *R.P.I Math Report. Rensselaer Polytechnic Institute, Troy, NY*, (No. 97-100):81–90.

[Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Cambridge University Press.

[Broomhead and Lowe, 1988] Broomhead, D. and Lowe, D. (1988). Multivariable functional interpolation and adaptative networks. *Complex Systems*, 2:321–355.

[C.G.Atkenson et al., 1997] C.G.Atkenson, A.W.Moore, and S.Schaal (1997). Locally weighted learning. *Artificial Intelligence Review*, 11:11–73.

[Chinrungrueng and Séquin, 1995] Chinrungrueng, C. and Séquin, C. (1995). Optimal adaptative k-means algorithm with dynamic adjustment of learning rate. *IEEE Transactions on Neural Networks*, 6:157–169.

[Cover, 1965] Cover, T. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326–334.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to support vector machines.* Oxford: Clarendon Press.

[Dasarathy, 1980] Dasarathy, B. (1980). Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed enviroments. *Patterns Analysis and Machine Intelligence*, 2:67–71.

[de Carvalho and Brizzotti, 2001] de Carvalho, A. and Brizzotti, M. (2001). Combining rbf networks trained by different clustering techiques. *Neural Proccessing Letters*, 14:227–240.

[Duda and Hart, 1973] Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* Wiley-Interscience Publication, New York.

[Friedman, 1994] Friedman, J. H. (1994). An overview of predictive learning and function approximation. *From Statistics to Neural Networks, Proc NATO/ASI Workshop, Springer Verlag*, pages 1–61.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmit, M. (1997). Bayesian networks classifiers. *Machine Learning*, 29:131–163.

[Fritzke, 1994] Fritzke, B. (1994). Fast learning with incremental rbf networks. *Neural Processing Letters*, 1:2–5.

[Gates, 1972] Gates, G. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:431–433.

[Geman et al., 1992] Geman, S., Bienenstok, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

[Ghosh and Nag, 2000] Ghosh, J. and Nag, A. (2000). *An Overview of Radial Basis Function Networks.* R.J. Howlett and L.C. Jain (Eds). Physica Verlag.

[Hart, 1968] Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516.

[Hassibi and Storkt, 1993] Hassibi, B. and Storkt, D. (1993). Second order derivatives for network prunning: optimal brain surgeon. *Advances in Neural Information Processing Systems. Morgan kaufman*, 5:164–172.

[Haykin, 1999] Haykin, S. S. (1999). *Neural networks : a comprehensive foundation.* Prentice Hall.

[Ismail and Kamel, 1989] Ismail, M. and Kamel, M. (1989). Multidimensional data clustering utilizing hybrid search strategies. *Pattern Recognition*, 22:75–89.

[Ismail et al., 1984] Ismail, M., Selim, S., and Arora, S. (1984). Efficient clustering of multidimensional data. *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*, pages 120–123.

[Jang and Sun, 1993] Jang, J. and Sun, C. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159.

[Kadirkamanathan and Niranjan, 1993] Kadirkamanathan, V. and Niranjan, M. (1993). A function estimation approach to sequential learning with neural network. *Neural Computation*, 5:954–975.

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. *Proc. IEEE*, 78(9):1464–1480.

[Kohonen, 1995] Kohonen, T. (1995). *Self-organizing maps.* Berlin: Springer.

[Krzyzak and Linder, 1997] Krzyzak, A. and Linder, T. (1997). Radial basis function networks and complexity regularization in function learning. *Advances in Neural Information Processing Systems*, 9:197.

[Kubat, 1998] Kubat, M. (1998). Decission trees can initialize radial basis function networks. *IEEE transactions on Neural Networks*, 9:813–821.

[Leonardis and Bischof, 1998] Leonardis, A. and Bischof, H. (1998). An efficient mdl-based construction of rbf networks. *Neural Networks*, 11:963–973.

[Light, 1992a] Light, W. (1992a). Ridge functions, sigmoidal functions and neural networks. *Aproximation Theory VII*, pages 163–206.

[Light, 1992b] Light, W. (1992b). Some aspects of radial basis function approximation. *Aproximation Theory, Spline Functions and Applications*, 256:163–190.

[Lloyd, 1982] Lloyd, S. (1982). Least square quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137.

[Lowe, 1989] Lowe, D. (1989). Adaptative radial basis function nonlinearities, and the problem of generalisation. *First IEE International Conference on Artificial Neural Networks*, pages 171–175.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, 1:281–297.

[Markovich and Scott, 1989] Markovich, S. and Scott, P. (1989). Information filters and their implementation in the syllog system. *Proceedings of the Sixth International Workshop on Machine Learning. Ithaca, NY. Morgan Kaufmann*, pages 404–407.

[Mhaskar, 1996] Mhaskar, H. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8:1731–1742.

[Michelato et al., 1983] Michelato, A., Mosetti, R., and Viezzoli, D. (1983). Statistical forecasting of strong surges and aplication to the lagoon of Venice. *Boll. Ocean. Teor. Appl.*, 1:67–83.

[Michelli, 1986] Michelli, C. (1986). Interpolation of scattered data: Distances matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning.* McGraw-Hill Series in Computer Science.

[Moody, 1994] Moody, J. (1994). Prediction risk and architecture selection for neural networks. *From Statistics to Neural Networks, Proc. NATO/ASI Workshop. Springer Verlag*, pages 143–156.

[Moody and Darken, 1989] Moody, J. and Darken, C. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294.

[Moore and Lee, 1994] Moore, A. and Lee, M. (1994). Efficient algorithms for minimizing cross validation error. *Proceedings of the 11th International Conference on Machine Learning. San Francisco.*

[Mulgrew, 1996] Mulgrew, B. (1996). Applying radial basis functions. *IEEE Signal Processing Magazine*, 1:50–65.

[Niyogi and Girosi, 1996] Niyogi, P. and Girosi, F. (1996). On the relationship between generalization error, hypothesis complexity and sample complexity for radial basis functions. *Neural Computation*, 8:819–842.

[Orr, 1995] Orr, M. J. L. (1995). Regularization in the selection of radial basis function centers. *Neural Computation*, 7:606–623.

[Orr, 1996] Orr, M. J. L. (1996). Introduction to radial basis neural networks. *Technical Report. Centre for Cognitive Science, University of Edinburgh.*

[Park and Sandberg, 1991] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257.

[Park and Sandberg, 1993] Park, J. and Sandberg, I. W. (1993). Universal approximation and radial-basis-function networks. *Neural Computation*, 5:305–316.

[Platt, 1991] Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225.

[Poggio and Girosi, 1990] Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proc. IEEE*, 78:1481–1497.

[Powell, 1985] Powell, M. (1985). Radial basis function for multivariate interpolation: a review. *IMA Conference on Algorithms for the Aproximation of Functions and Data*, pages 143–167.

[Powell, 1988] Powell, M. (1988). Radial basis function approximations to polynomials. *Numerical Analysis 1987 Proceedings*, pages 223–241.

[Rissanen, 1984] Rissanen, J. (1984). Universal coding, information, prediction and estimation. *IEEE Transactions on Information Theory*, 30:629–636.

[Scholkopf et al., 1998] Scholkopf, A., Burges, C., and Smola, A. (1998). *Advances in kernel methods-support vector learning.* MIT Press.

[Schwenker and Dietrich, 2000] Schwenker, F. and Dietrich, C. (2000). Initialization of radial basis function networks by means of classification trees. *Neural Network World*, 10:473–482.

[Schwenker et al., 2001] Schwenker, F., Kestler, H., and Palm, G. (2001). Three learning phases for radial basis function networks. *Neural Networks*, 14:439–458.

[Schwenker et al., 1994] Schwenker, F., Kestler, H., Palm, G., and Höher, M. (1994). Similarities of lvq and rbf learning. *Proc. IEEE International Conference SMC*, pages 646–651.

[Sejnowski and Rosenberg, 1987] Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

[Shim and Cheung, 1995] Shim, C. and Cheung, J. (1995). Pattern classification using rbf based fuzzy neural networks. *Intelligent Engineering Systems Through Artificial Neural Networks*, 5:485–490.

[Ster and Dobnikar, 1996] Ster, B. and Dobnikar, A. (1996). Neural networks in medical diagnosis: Comparison with other methods. *Proceedings of the International Conference EANN '96*, pages 427–430.

[Tikhonov and Arsenin, 1977] Tikhonov, A. and Arsenin, V. (1977). *Solutions of ill-posed problems.* W.H. Winston.

[Tomasin, 1973] Tomasin, A. (1973). A computer simulation of the Adriatic Sea for the study of its dynamics and for the forecasting of floods in the town of Venice. *Comp. Phys. Comm.*, 5:51.

[Vapnik, 1992] Vapnik, V. (1992). Principles of risk minimization for learning theory. *Advances in Neural Information Processing Systems*, 4:831–838.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory.* New York: John Wiley and Sons.

[Wettschereck et al., 1997] Wettschereck, D., Aha, D., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314.

[Wettschereck and Dietterich, 1992] Wettschereck, D. and Dietterich, T. (1992). Improving the perfomance of radial basis function networks by learning center locations. *Advances in Neural Information Processing Systems*, 4:1133–1140.

[Whitehead and Choate, 1995] Whitehead, B. A. and Choate, T. D. (1995). Cooperative - competitive genetic evolution of radial basis function centeres and widths for time series prediction. *IEEE Transactions on Neural Networks*, 5:15–23.

[Xu et al., 1994] Xu, L., Krzyzak, A., and Yuille, A. (1994). On radial basis function nets and kernel regression: Statistical consistency, convergence rates and receptive field size. *Neural Networks*, 7:609–628.

[Yingwei et al., 1997] Yingwei, L., Sundararajan, N., and Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478.

[Zaldívar et al., 2000] Zaldívar, J., Gutiérrez, E., Galván, I., Strozzi, F., and Tomasin, A. (2000). Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks. *Journal of Hydroinformatics*, 2:61–84.

[Zhang et al., 1997] Zhang, J., Yim, Y.-S., and Yang, J. (1997). Intelligent selection of instances for prediction functions in lazy learning algorithms. *Artificial Intelligence Review*, 11:175–191.